POLITECNICO DI TORINO

Master's Degree in MECHATRONIC ENGINEERING

Master's Degree Thesis

Terrain Traversability Analysis for Planetary Exploration Rovers

Supervisors

Prof. MARCELLO CHIABERGE

FRANCESCO MESSINA

GIACOMO FRANCHINI

Candidate

Achille CHIUCHIARELLI

JULY 2024

Abstract

The field of autonomous navigation for unmanned ground vehicles (UGVs) is in continuous growth and increasing levels of autonomy have been reached in the last few years. The task becomes, however, more challenging when the focus is the exploration of planet surfaces, such as Mars. In those situations, UGVs are forced to navigate through unstable and rugged terrains which, inevitably, open the vehicle to more hazards, accidents, and, in extreme cases, complete failure of the mission. This thesis tackles the problem of terrain traversability analysis in the context of planetary exploration rovers, delving particularly into Mars exploration. The aim of the research is the development of a hybrid architecture, which enables the assessment of terrain traversability evaluating the results of both an appearance-based approach and a geometry-based approach. The coexistence of the two methodologies has the objective of balancing each other's flaws, reaching a more robust and complete understanding of the operating environment. The appearance-based method employs semantic segmentation, operated by a deep neural network, to understand the different classes of terrain present in the scene. Predictions are refined by an additional module that performs pixel-level terrain roughness classification from the same RGB image. The rationale behind this choice resides in the will to be able to assign different costs, even to areas belonging to the same terrain class, while including an analysis of the physical properties of the soil. This first cost map is then combined with a second one yielded by the geometry-based approach. This module evaluates the geometrical characteristics of the surrounding environment, highlighting categories of hazards that are not easily detectable by semantic segmentation. The proposed architecture has been trained using synthetic datasets and developed as a ROS2 application to be easily integrated into a higher-level framework for autonomous navigation in harsh environments. Simulations have been performed, showing the ability of the method to assess online traversability analysis.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Context overview

Space exploration has been one of the biggest goals of humankind, starting from the 50_s to present days, pushing further and further the boundaries of what it is possible to achieve. Among the many routes through which this field branches, a topic always under the spotlight is planetary exploration.
Beginning from the race to the Moon, undertaken by the United States and the Soviet Union during the years of Cold War, the will to explore celestial bodies, different from the Earth, has never faded away and, after the Moon has been conquered, the focus rapidly moved towards the red planet, Mars. It is exactly in this context that this thesis work puts its bases, presenting a possible framework for the terrain traversability analysis of a planetary exploration rover autonomously navigating through the Martian surface.

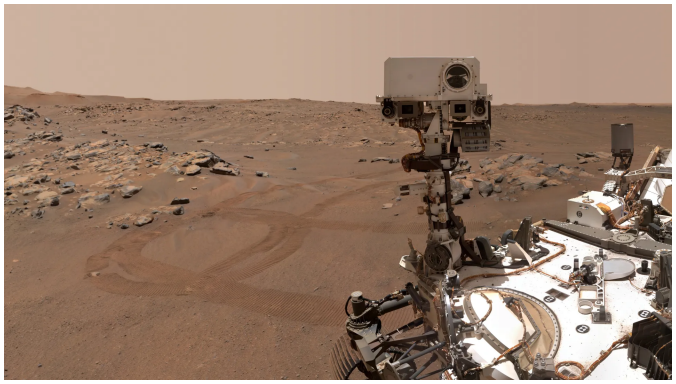

Figure 1.1: NASA mars rover Perseverance.

Mars represents the next step in the challenges faced by humans, it holds a lot of new possibilities:

- more reasonable, such as the quest for new information, to make our knowledge about the universe advance, or for the resources which could be found on the planet;
- more "fictional", such as finding extraterrestrial life or creating human bases on its surface (1.2).

However, it is still not even possible to easily achieve human presence in missions, as stated by Michelle Rucker, lead of NASA's Human Mars Architecture Team, and reported in [1]. Tree are the main reasons behind this impossibility:

- **Distance**, Mars in its closest possible position is at 55 million km from the Earth and the most favourable window to embark on this mission happens every 26 months, or if it is looked for the absolutely best possible conditions every 15 years. Other than this inconvenient, an important role to actually travel the distance will be determined by the propulsion system of the employed spaceship. To be able to reach the planet quicker and more often, a new technology, most likely based on nuclear thermal propulsion or nuclear electric propulsion, would be required, but it is not available yet.
- **Living conditions**, supposing to have the vehicle capable of completing the traverse, another open question resides in the way human beings will react to living for several months on a spaceship with very limited space and low gravity.
- **Landing**, once the spacecraft has entered the orbit of the red planet, a new important obstacle to be faced is encountered, the act of landing on its surface. A novel deceleration system must be employed, to decrease the thrust of the vehicle and allow a safe landing. It would also be necessary to be prepared to resist to sand storms, once on the planet, and have all the equipment needed by the astronauts, already there before their arrival.

Figure 1.2: NASA martian base in the movie The Martian by Ridley Scott.

Being, still, so far from safe human missions, is of paramount importance to obtain as many information as possible on the planet, without the need of having a human operator involved and that's why mars rovers have been created. The environment of the planet is extreme: high radiations, really low temperatures, powerful storms; as a consequence, the machines sent to navigate through those harsh conditions need to be really robust and durable. For this reason, the development process of a rover is really long and requires a lot of resources.
Having depicted this setting, it is clear how fundamental is to grant that the navigation of the vehicle is as safe as possible and, being still not possible to control them in real time from the Earth, the key to obtain such safety is enclosed into the performances of its autonomous navigation. Autonomous control systems have improved exponentially due to the corresponding drastic growth of computational power of machines. As a result, the space missions have employed more and more those systems to achieve better performances, demanding increasingly more advanced technologies. At first, analog control has been replaced by digital control, then, the higher level of operations' complexity has required the introduction of machine learning techniques. This necessity has lead, eventually, to what is the current state of the art, a wide use of deep learning models, computer vision, neural networks and more.

1.1.1 General description of the project

This thesis work perfectly fits into the described landscape revolving around terrain traversability analysis, a topic characterized by significant contributions from the aforementioned technologies. This process features the study of the environment around the rover with the aim of gaining an understanding of the surroundings, finalized to the achievement of a safe and aware navigation without the need of human intervention. The information retrieved during this procedure are essential

in the avoidance of hazards which could damage the vehicle and endanger the correct progress of the mission.
The data can be acquired from several sources and represent diverse characteristics of the terrain. What will be presented, in this thesis, is the acquisition of information from a single source, a RGBD camera, with the aim to combine color and depth data in order to have a more complete understanding of the context. For what concerns the RGB images, the samples undergo a semantic segmentation process, trying to distinguish the different terrain classes in every picture. The objective of this operation is the employment of this knowledge in the assignment of a traversability cost, assessing which kind of soil is more suitable to be navigated and which one, instead, should be avoided if possible. The assignment of the costs is empirical and determined by the context and the kind of rover supposed to employ this framework, depending on the structural characteristics of the vehicle and the mission it is designed for. To increase further more the variables taken into account in the determination of the cost, another criteria is evaluated over the RGB images: a second classification process is operated to estimate the roughness values of the terrain depicted in each sample. The two obtained cost maps are then fused together to yield the final result of this section of the framework. To realize this, two different neural networks are required, which need to undergo a training process to gain the skills needed to perform the predictions. To fulfill this task and get to the final result, two different datasets have been employed:

- **AI4Mars**, a NASA dataset of real images used for the preliminary phases of the development.
- **Synthetic set**, a dataset of artificial images of a martian landscape used to achieve the goal of creating a unique set to train both networks. The creation of those samples has become necessary to retrieve all the needed information to train the roughness classification, due to the lack of depth data in real images sets.

Figure 1.3: Example of semantic segmentation predicted mask over-imposed to the corresponding RGB image.

As introduced previously, the method studied in this work is a **Hybrid approach**, meaning that diverse information are integrated together. Other than the RGB images, also depth images have been employed to asses traversability. In this case, the evaluation is based on the analysis of the geometrical features of the terrain. The traversability cost is here assigned as a function of how intense are the geometrical alterations in the area covered by the range sensor. A lower cost will be assigned to surfaces more plane and clear, while regions with abrupt changes in the slope of the terrain or presenting obstacles will receive higher costs.
Once the cost maps, obtained with the two approaches, have been computed, the aim of the hybrid approach is to combine them to obtain a final, overall, traversability map. The final product will be used by a path planner to make terrain-aware decisions, trying to avoid a series of possible hazards which made several missions fail.

1.2 Project structure

This thesis work is articulated into seven chapters, guiding throughout the process which leads to the complete framework for the analysis of terrain traversability:

- In **Chapter 2**, a comprehensive background of the history of planetary exploration, revolving around Mars, is presented to have an overview of the steps and events leading to the introduction of Unmanned Ground Vehicles as main actors of the aforementioned explorations. The evolution in complexity and technology of the rovers is displayed, using some of the most relevant Martian missions as representative examples. The chapter, then, ends with a brief summary of the fundamental building blocks of the machine learning tasks involved in the development of the framework.
- In **Chapter 3**, a review of the literature available on the core topics of the research is provided. In this list, the papers employed in the different stages of the work are reported and analyzed, leading to the choices made with respect to the structure of the framework itself and the most interesting approaches.
- In **Chapter 4**, the process to get to the final framework is eviscerated. All the steps in the development of the different modules of the framework are presented, explained in detail and a general description of their implementation is provided. At the end of the chapter the overall structure, joining together all the previously described blocks, is displayed and the communications between them are detailed.
- In **Chapter 5**, the phases of the whole development are contained together with the corresponding results achieved. Starting from the early tests of the singular modules, displaying the improvements obtained through those experiments, to the final simulation tests, providing a complete report of the different trials on the finished framework. In the end, an analysis of the performances obtained and a retrospective on the achieved goals is finally performed .
- In **Chapter 6**, to end this thesis work, the conclusions on all the research done and enclosed in the previous chapters are drawn and some possible future improvements to the presented result are proposed.

Chapter 2

Background

In this chapter will be presented an overview of the context in which this thesis work locates and of the theoretical concepts needed to understand properly the topics covered in the development of the framework. Starting from a comprehensive introduction on the history of the exploration of the red planet, some of the most important rovers' missions on its surface are presented and both their achievements and the reasons behind some of their failures are displayed. To finish the chapter, after a dive into the next steps of the evolution in rovers' technology, an outline of the main theoretical themes over which the foundations of the presented framework are laid is provided.

2.1 Mars exploration history

The history revolving around Mars has its roots in the second half of the 16^{th} century when Danish astronomer Tycho Brahe was the first ever to map accurately the movement of Mars across the sky. After him, the next historic progress we encounter is the first ever observation of Mars through a telescope made by Italian astronomer Galileo Galilei in 1610, followed by even more precise observations made by the Dutchman Christiaan Huygens. He was able to see a darker spot on the planet's surface which is attributed to be the region Syrtis Major. After these events, the studies about the red planet kept increasing until, during 1840, the first ever map of Mars (fig.2.1) was published by Johann Heinrich von Mädler and Wilhelm Beer. The next step in the history related to this planet is again a map, but the one published by Italian scientist Giovanni Schiaparelli in 1877 (fig.2.2). The latter needs to be particularly highlighted since is the one on which current nomenclature is based of and is also the one which provided to create the first ever fake news about Martian life, this was due to a mistranslation of the word "canali" into canals instead of channels, implying some kind of artificial.

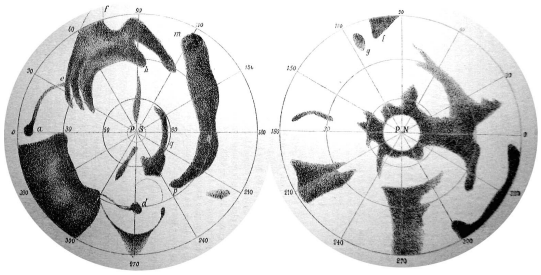

Figure 2.1: First map of mars published byJohann Heinrich von Mädler and Wilhelm Beer.

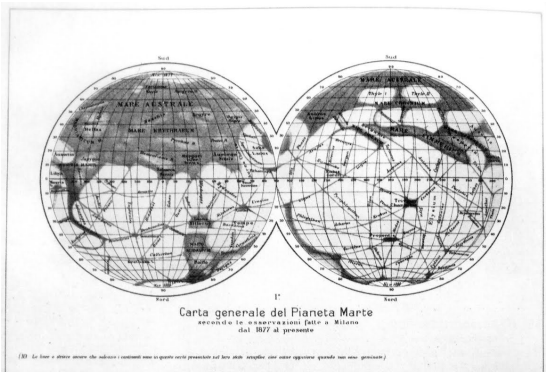

Figure 2.2: Mars map by Giovanni Schiaparelli.

The misunderstanding was definitively set aside when the Greek astronomer Eugene Antoniadi in 1909 observed that those "canals" where nothing but an optical illusion. The same Antoniadi is the one remebered to have published the most detailed maps of mars prior to space exploration (fig.2.3).

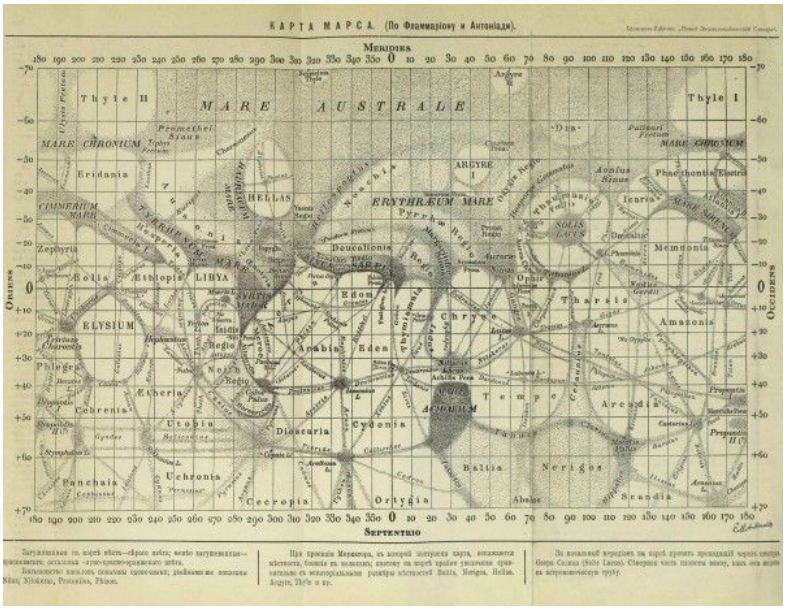

Figure 2.3: One of the mars maps published by Eugene Antoniadi.

The next big leap in space exploration and in particular planetary exploration, was, obviously, the beginning of the era of space travels.

The reasons behind the interest in the exploration of Mars are not only the same fueling space exploration in general, so the interest in understanding how the universe works, how celestial bodies were formed and how the Earth origined, but also more tactical reasons. Mars, among the other planets, has been proved to be the one with the best conditions in terms of compatibility of atmospheric structure, visibility, durability of machines and possible renewable sources to extend mission's life. Obviously, as a consequence of the progresses made in those years, the next logical step has been to send remotely controlled objects, which would have allowed more control over the explorations. It was realized, for the first time, in 1998 with the Sojourner Rover. The machine couldn't travel on the surface but was capable of sending 16500 images to Earth. This feat was the one which contributed to rekindle, even more, the interest in the study and exploration of the 4^{th} planet of our Solar System, which lead in 2001 to a series of new historical missions:

- **Mars Odyssey** in 2001 (named after the Stanley Kubrick's masterpiece), the currently longest surviving continually active spacecraft in orbit around a planet other than Earth. Its mission is to search present or past evidences of water and ice, study the geology and radiation of the planet and serve as a relay for communications between the rovers and Earth.

- **Mars Exploration Rover** in 2004 with the famous twin rovers Spirit and Opportunity, which were capable of operating for far longer than what was initially intended for them. Spirit ended its mission officially in 2011, after being stuck in an irrecoverable way in a sand trap for two years (fig.2.4). Opportunity kept working until 2019 when NASA declared its mission to be over since the rover had been damaged during a sandstorm and was not possible to contact him ever since. Opportunity (fig.2.5), in particular, reached outstanding results since was able to find elements of water, source of potential microbial life.

- **Mars Science Laboratory** in 2012, with the purpose of studying the habitability of the red planet, including its climate and geology, and collecting data to understand if Mars ever had the right environmental conditions to support microbial life and to have more information for a future human mission. The first of the two rovers, which are part of this mission, is Curiosity (fig.2.6); a car sized rover which is carrying the most advanced instruments for scientific studies ever sent to the Martian surface and its exploring mission, at the end of 2012, has been extended indefinitely. Its companion is the Perseverance rover (fig.2.7), landed in 2021, with its helicopter drone Ingenuity, recently declared no more operational due to a reported damage in one of its blades. Its mission is, again, to search for clues of microbial life, but also to drill some samples of rock and regolith and store them for a possible return to Earth.

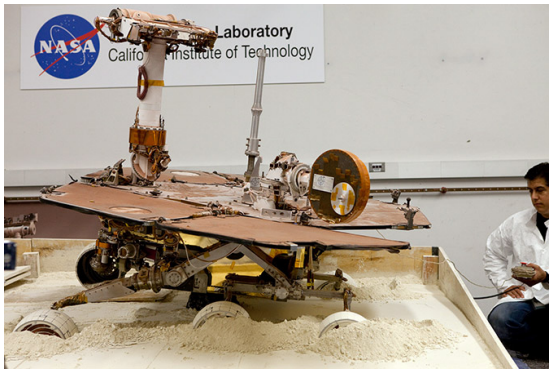

Figure 2.4: Reproduction made by NASA scientists of how Spirit was stuck in sand.

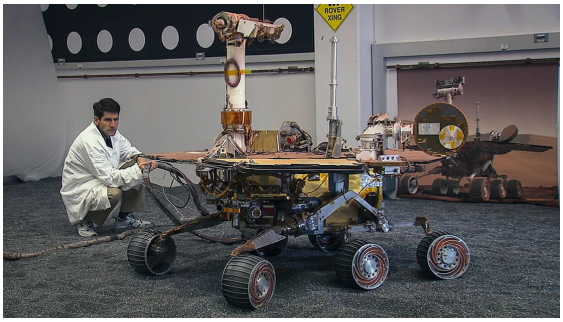

Figure 2.5: Opportunity rover.

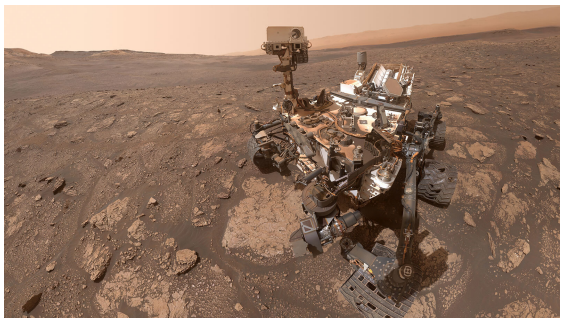

Figure 2.6: Self portrait of Curiosity rover.

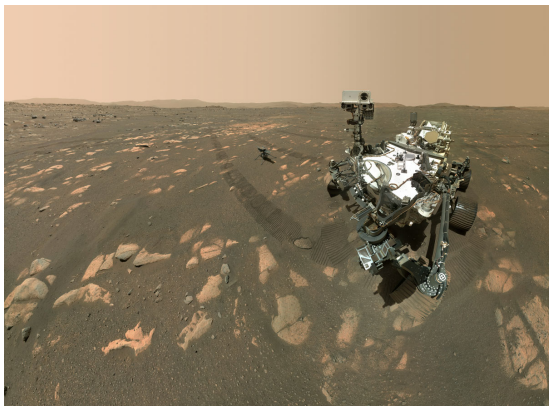

Figure 2.7: Self portrait of Perseverance and its companion Ingenuity.

2.2 Rovers' evolution

The exploration of Mars has witnessed a captivating evolution of robotic rovers, driven by advancements in mechatronic engineering and space technology, with a particular emphasis on the transformation of navigation capabilities. The start of this journey, as we already mentioned, took place in 1997 with the deployment of the Sojourner rover, marking the first successful attempt into remotely operated Martian exploration. The evolution of Mars rovers has since been characterized by significant progresses in autonomous navigation, transforming these machines from basic mobility platforms into sophisticated, self-reliant scientific instruments capable of executing complex tasks.

The technological advance of Mars rovers has been strongly tied to the iterative process of learning, with each mission building upon the successes and improving from the failures of its predecessors. The twin rovers Spirit and Opportunity, exemplified a leap forward in autonomous navigation capabilities. They showcased improved autonomy, allowing them to navigate the Martian terrain with greater independence. Spirit and Opportunity's onboard systems were equipped with advanced algorithms and sensors, enabling them to make real-time decisions about their path, avoiding obstacles and optimizing travel efficiency [2]. This shift towards increased autonomy significantly reduced reliance on Earth-based teleoperation, enhancing the rovers' ability to adapt to new challenges on the Martian surface.
A crucial milestone in autonomous navigation was achieved with the Mars Science Laboratory presentation of the Curiosity rover. Curiosity represented a paradigm shift, not only in size and scientific capabilities, but also in the rover's ability to autonomously traverse the Martian landscape. The navigation system incorporated advanced algorithms and hazard avoidance mechanisms, allowing it to identify and navigate around obstacles in real-time. Curiosity's autonomous capabilities paved the way for more efficient exploration, allowing the rover to cover greater distances and respond dynamically to the diverse characteristics of the Martian terrain [3]. The latest advancement in autonomous navigation is represented by the Perseverance rover, equipped with cutting-edge mechatronic technologies [4], including an advanced mobility system that enables more precise and agile movements. The rover's autonomous navigation system, guided by powerful onboard software (SPOC) [5] and enhanced sensors, grants it the skills to make informed decisions about traversing challenging terrains. This increased level of autonomy is crucial for achieving mission objectives, as Perseverance can efficiently navigate to specific points of interest and execute scientific experiments without constant Earth-based intervention.

Looking forward, the evolution of autonomous navigation in Mars rovers remains

a focal point for future missions. The European Space Agency's (ESA) Rosalind Franklin rover (fig.2.8), yet to be launched, embodies the next frontier in mechatronic engineering applied to planetary exploration. With a new suite of instruments designed for autonomous operations, including advanced hazard detection and avoidance capabilities, it promises to push further the boundaries of what is achievable in Martian surface exploration. In particular, it is set to be able to improve the behaviour of the rover in situations which were hazardous for its predecessors (e.g. Spirit and its accident). This, in practice, will be granted by a new locomotion mode, called "wheel walking", allowing the articulation of motion around the axes and adjusting the rover height and angle with respect to the surface. This motion is supposed to give very good traction in soft soils and high slopes, such as dunes. With its stereo vision system, it will be capable of identify obstacles and slopes and also take into account wheel slippage [6]. The goal is to have the rover traversing autonomously and safely for approximately 100 m per sol (martian day).

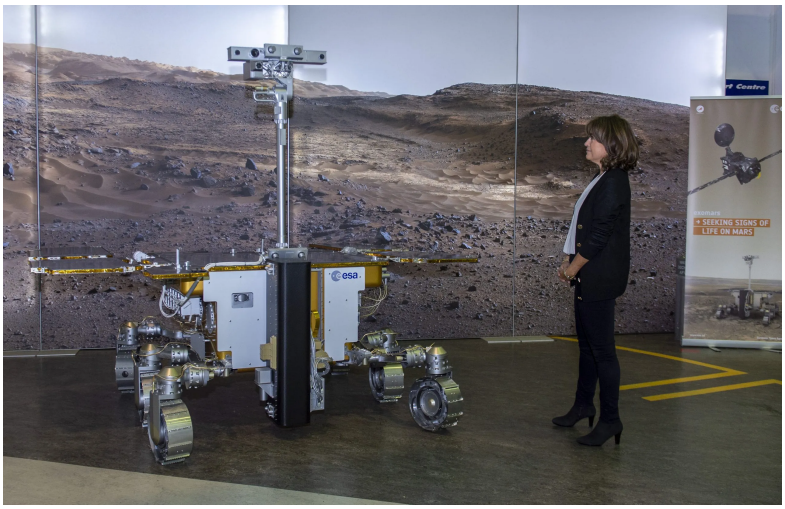

Figure 2.8: Rosalind Franklin rover.

2.3 Terrain traversability analysis

The topic of autonomous navigation is strictly linked with the topic of traversability analysis. For what concerns off-road environments, class which comprises planetary exploration, it revolves around the traversability of the terrain. The agent, in these kind of settings, needs to analyze the traversability of the surrounding surface and plan an optimal and landscape-aware path according to it. The causes of this need have to be found in the conditions, these kind of environments, impose on the vehicles, the driving surface is uneven, not consistent and can present obstacles and potential hazards which could be cause of damages leading, in extreme cases, to the complete failure of the mission. Terrain analysis is defined as the problem of estimating the difficulty of driving through a terrain for a ground vehicle [7]. In order to accomplish this task, it is needed the use of several areas of machine learning which make the UGV (Unmanned Ground Vehicle) capable of autonomously obtain all the information it needs to fulfill traversability analysis and be as aware as possible of what surrounds it, when planning the path to follow.

2.4 Deep learning and computer vision

2.4.1 Deep learning

In the first half of the 20^{th} century, science fiction familiarized the world with the concept of artificially intelligent robots and by the 1950s a generation of scientists had already assimilated the concept of AI into their minds [8]. One of these illustrious minds was Alan Turing, the first to suggest and mathematically explore the possibility of artificial intelligence. His belief was that, if humans were capable of using available information to reason and solve problems, why couldn't machines do the exact same thing. He went beyond just having the intuition, in fact, in his 1950 paper "Computing Machinery and Intelligence" he presented a proposal on how to build intelligent machines and test their reasoning.
The first artificial intelligence program, called Logic Theorist, was presented in 1956 and was meant to mimic the problem solving skills of a human. It was a milestone in the history of the field since, after it, everyone was finally convinced that AI was achievable. When we talk about Machine learning, term coined in 1962 by Arthur Samuel an employee of IBM, we are discussing about a branch of artificial intelligence which is aiming at imitating the learning capabilities of the human brain to learn by gradually improving the accuracy of what it is required to do. This is done through an iterative process, called training, during which a neural network, a machine learning model that is built using principles of neuronal organization, is continuously trying to decrease the difference between its predictions and the actual outputs of a given dataset. The two main uses of machine learning are:

- **Classification** of data based on models which have been developed.
- **Predictions** of future outcomes based on these models.

With the exponential increase in computers' capabilities and the research for ways to process inputs with more and more complex architectures, technology was pushed to the point where simple machine learning models were not sufficient to work with these data and were not capable to extract the needed features. For these reason a new branch of AI grew in popularity, Deep Learning. As Russian-American computer scientist Lex Fridman defined, deep learning can be considered as "scalable machine learning", so it starts were machine learning is, but focuses on the use of multi-level architectures, which were developed to process deeper features from information given as input.
As mentioned earlier, the whole learning aspect revolves around the use of neural networks which can be briefly described as structures made of linked neurons, their building blocks. These neurons are grouped into layers and each layer is characterized by a specific number of neurons and activation functions. They are special kinds of functions which are used to derive an output from a set of input

values fed to a node (or a layer) with the purpose to add non-linearity to the network. They change depending on the application the network is defined for, but they are always needed. Otherwise, even a network with lots of neurons and layers could be easily described as 1 neuron which would not be capable to extract deep features from the inputs we provide it. The first layer of a network is called input layer, the last one is called output layer, while the in-between layers are all called hidden layers. When the network is made by more than three layers it can be defined as a Deep Neural Network (DNN). Deep learning, other than allowing to work with more complex and unstructured data like text and images, also automates feature extraction. In deep learning models is the network itself to decide which features are most important to fulfill its task, which in machine learning is instead established manually by a human expert.

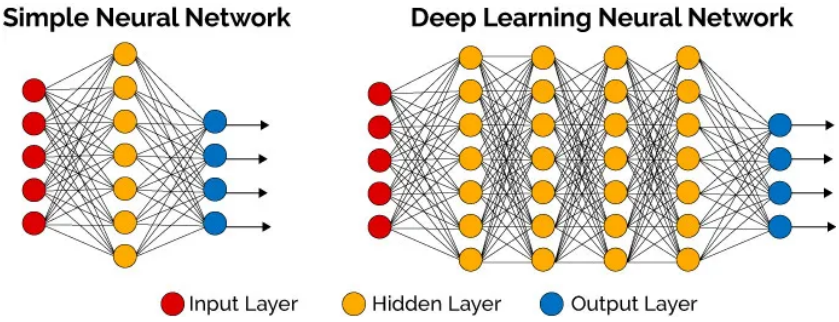

Figure 2.9: Comparison between simple neural networks and deep neural networks' architectures.

2.4.2 Computer vision

When discussing about terrain traversability analysis, the information needed to study the surroundings can come from several sources which could be internal sensors of the rover, e.g. accelerometers or wheel slippage, but also external to the vehicle and gathered from sensors like lidars or cameras. Whenever the information is derived from any kind of visual inputs, we are talking about a really important and diverse field of artificial intelligence which is called Computer Vision. What we are doing, when working with a CV task, is, in practice, enabling computers to see and to take actions or make recommendations based on what they are seeing. To be able to complete these tasks, the neural network is fed with a large quantity of visual data and it keeps running analyses on those information until it discerns distinctions and, ultimately, recognizes images. The aim of the field is to be able to disentangle symbolic information from image data using models constructed with the aid of geometry, physics, statistics, and learning theory.

In the current landscape of research and industry, computer vision is vastly applied and is employed in a wide range of fields. Among the most important ones it is possible to find:

- **Space exploration**, to mention just a couple of the uses in this area we may think of autonomous navigation of rovers, but also docking operation of spacecrafts.
- **Automotive**, most of the ADAS systems in the field are based upon computer vision and visual inputs.
- **Medical applications**, the analysis of medical imagery highly relies on computer vision and, in some cases, they even surpassed human performances.
- **Industrial production lines**, applications like object detection, measurements or autonomous navigation of carts are widespread in modern facilities.
- **Agriculture**, it could seem a bit strange, but some really important tasks can benefit from the contribute of computer vision applications in this area. Some examples being, analysis of the terrain, either its health status or its composition, or harvesting operations which are becoming more and more automatic.

2.4.3 Convolutional neural networks

The main actors, when it comes to computer vision applications, are Convolutional Neural Networks, or CNNs. This kind of networks, inspired by the organization of the animal visual cortex, as suggested by the name, relies on the use of particular layers called convolutional layers, which, how it is easily deducible, operate convolution on their inputs. These layers are needed in image processing due to the extremely high number of features coming from even a single RGB image; considering that, with normal resolution, average dimensions would be 3x512x512 we would have 786432 features to analyze. Convolutions are advantageous in this sense since are operations able to produce a filtered image with reduced dimensions. Convolution, in image processing, is a kind of local averaging technique which uses a, so called, filter as weight pattern. A filter, or kernel, is a smaller matrix, compared to the input image (a widely used size is 3x3 for example), which is used as a sliding window to compute the, previously mentioned, weighted local average in portions of the image. Each step, a window of pixels is considered and the output of this operation will be a single pixel of the new filtered matrix, its value is obtained by taking the one of the central pixel and adding the weighted values of all its neighbors together. After sliding the filter over the whole picture, the output of the convolution is obtained and it consists of a new filtered image, with

smaller dimensions, containing the extracted features. Every layer can have several different filters and they are what will be trained during the training procedure for convolutional layers.

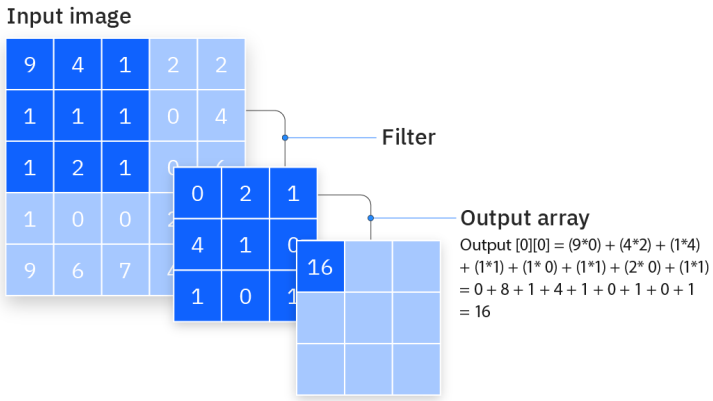

Figure 2.10: Convolution operation.

Other than convolutional layers, the characteristic architecture of CNNs contains activation layers and pooling layers. The first one is used to break linearity of the filtered feature map and the latter to reduce, with a pooling operation, the spatial size of the map. The process is usually repeated several times to extract deeper and deeper features.

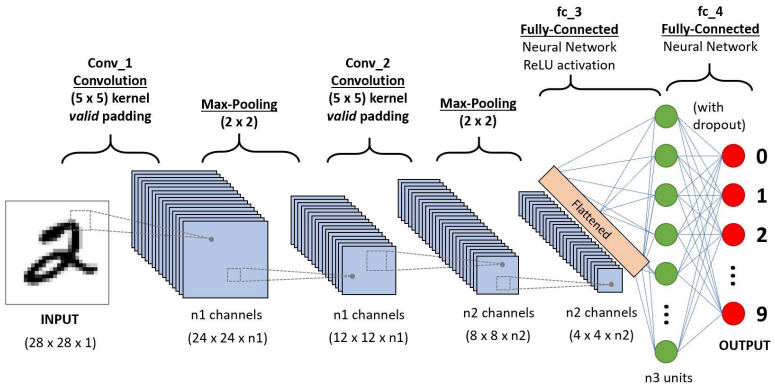

Figure 2.11: Example of the architecture of a CNN.

Among the variety of tasks which can be addressed using this kind of neural networks, this thesis work focuses mainly on Semantic segmentation and roughness classification of the acquired images.

2.4.4 Semantic segmentation

Semantic segmentation is, again, a computer vision task with the aim of assigning to each pixel of an image a label corresponding to a certain class. It falls in the wider category of Image Segmentation, in which are present also Instance and Panoptic segmentation, so the analysis process of an image, section by section, and the classification of the information contained in each portion. All the three mentioned processes use different criteria for the classification of each pixel in the image, taking advantage of information like color, contrast, placement within the picture and other attributes.
Semantic segmentation, in particular, creates a, so called, segmentation map. In this map every pixel is assigned to a specific class and is accordingly color coded, resulting, in this way, in an understanding of the whole scene represented by the image. The classes can be various and they are related to the particular task the segmentation is used for; it could be used to recognize foreground and background pixels, specific categories of objects like cars, buildings and so on or, as it will be seen in the following chapters, to differentiate the semantic characteristics in terrain images highlighting the different kinds of terrain a vehicle is surrounded by. One of the most commonly used and efficient architectures, for deep neural networks used to achieve semantic segmentation, is the encoder-decoder model (fig.2.12).

- **Encoder**: is defined as the portion of the network responsible for the processing of the input data. Followed by the extraction of the high level and abstract features, called hierarchical features, and the capture of contextual information, like relationships between different objects in the image. It is done involving a series of convolutional layers reducing the spatial dimensions while increasing the depth of feature maps.

- **Decoder**: consists of the portion of the network devoted to retrieve, from the compact and abstract representation generated by the encoder, an output that has the same spatial dimensions as the original input. This process is necessary to recover the spatial information lost during the downsampling of the encoder and obtain back a high resolution output which enables pixel-wise classification. These operations are done using some kind of upsampling of the feature maps through techniques like transposed convolution.

The connection between these two elements is, also, of paramount importance since acts as bridge between the high-level abstract features learned by the encoder and the spatially detailed features needed by the decoder. Additionally, it is common to find, in these kinds of model, further links between corresponding layers of encoder and decoder, called skip connections. They serve as a way, for the decoder, to access the features from the lower levels of the encoder which could help in improving performances of the network with respect to the vanishing gradient problem. The

latter, is a phenomenon, which can be encountered during the training procedure of deep neural networks, consisting in the gradients, used to update the network, becoming extremely small, so almost vanishing.

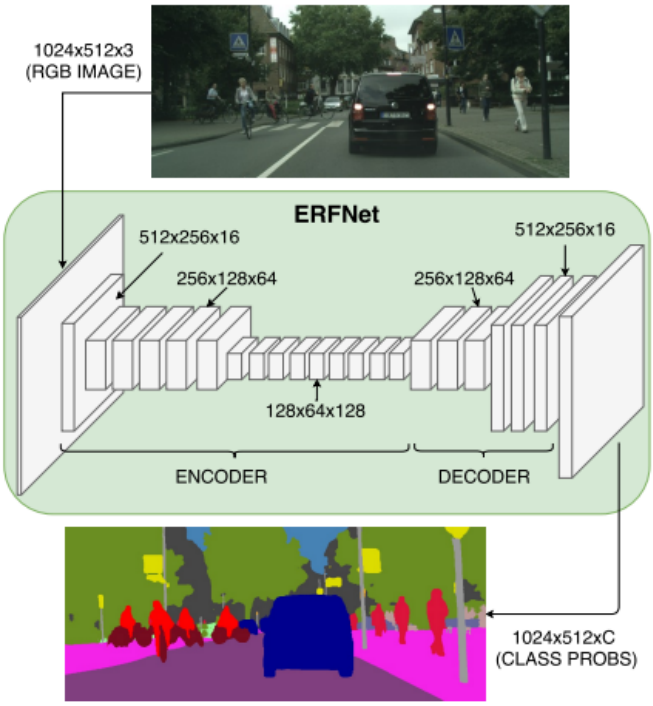

Figure 2.12: Encoder Decoder convolutional network used for semantic segmentation [9].

2.4.5 Regression

Discussing about semantic segmentation, it was mentioned the assignment of label values to each pixel of an image. Those labels are discrete numbers used as a code to assign not a value, but a class and, in fact, these kind of tasks are called **classification** problems. When, instead, the aim is to predict a continuous number, or value, the type of the problem, which is being considered, is **regression**. The objective of a regression task is to analyze the input data and to map them into a continuous space which could represent many different kind of quantities, temperatures, coordinates etc. The choices of architecture of the neural network, loss function, and evaluation metrics can be various and depend on the specific characteristics of the considered regression task.
In the field of computer vision, focus of this thesis work, several applications benefit from regression tasks among which we may find:

- Depth estimation from monocular RGB images.
- Pose estimation of an object (e.g. a robotic manipulator).
- Prediction of facial key-points.
- Prediction of a physical characteristic of the terrain in an image [10] (as will be mentioned later on).

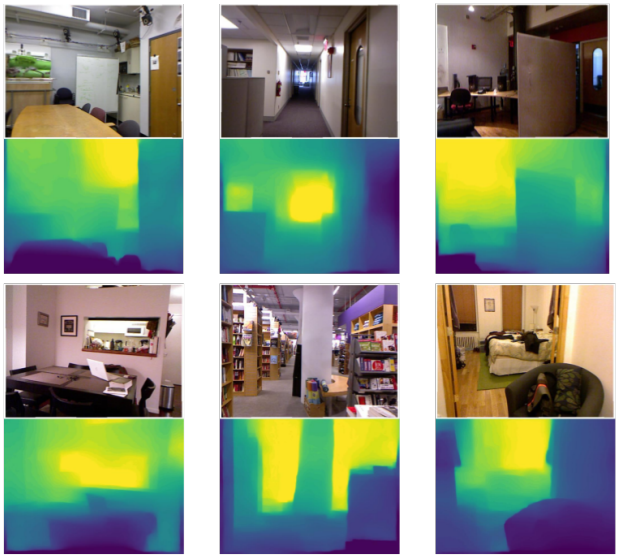

Figure 2.13: Examples of monocular depth estimation solving a regression problem[11].

Chapter 3

Literature review

In this chapter, as suggested by the title, a dive into the literature published over the years regarding the topic of terrain traversability analysis and correlated subjects, fundamental for the development of the framework, is presented. After exploring the problem's definition, the different paths followed by previous researchers are described and the choices behind the structure and the methodologies employed in this framework are defined.

3.1 Problem's description

Discussing about the field of robotic applications, side to side with indoor robots operating in structured environments, like companies' facilities, it can be found the field of outdoor robotics. The interest and need of deploying robots in off-road natural and unnatural environments is growing more and more as time goes by. Several are the applications which benefit from the use of UGVs to, either, aid human operators or even substitute them; among those, just to mention a few it is possible to find:

- **Forestry**, UGV equipped with advanced sensors and autonomous navigation algorithms can traverse really challenging terrains and obtain invaluable information. They can be employed for tasks like forests' mapping, surveillance, monitoring and many more [12]. An example of that is identifiable in [13], where a laser scanner, on an unmanned ground vehicle, is adopted to obtain a map of a such complex environment to the precision of signaling the position of single trees.

- **Mining**, in this field the use of UGVs has mainly two functions. The first one is propelling the exploration, in both subterranean and subaqueous operations,

while the second one is promoting a safe environment in which machines and humans work closely minimizing risks and reducing accidents [14].

- **Search and rescue** operations, when dealing with post-disasters intervention and, in particular for Urban Search and Rescue (USAR), ground robots have proved to be a very useful tool to take advantage of. Among the multiple benefits, the use of this kind of robots can produce, it can be mentioned a faster localization of potential victims, shorter rescue times, less risk for the human operators and faster assessment of damages to buildings [15]. Even though the vast majority of those vehicles work in a remotely controlled way, more and more research is being published to study the implementation of increasing levels of autonomy [16] [17].

- **Planetary exploration**. As previously, and extensively, mentioned in chapter 2, the use of UGVs in planetary exploration is fundamental. Being space exploration characterized by extreme conditions, high risks, really dilated times and enormous costs it is still not suitable for a growing presence of human operators and it finds a great aid in the employment of rovers. Those vehicles present advanced scientific instrumentation, allowing a broad range of possible missions to fulfill [18], and cutting-edge algorithms for autonomous navigation, making them able to traverse diverse planetary surfaces and be more and more independent from human intervention [5] [19].

A key point which links all of these, really different, fields is the need of the vehicles to adapt to the underlying terrain and, precisely, due to the wide range of working environments which the robot can face, it is possible to identify three common problems which need to be addressed to make operations feasible [20]:

1. Assessment of terrain traversability.

2. Planning optimal motion paths with respect to given criteria.

3. Suitable adaptation of the kinematic configuration of articulated robots as function of terrain traversability

Even if the three topics are highly interdependent, due to their complexity they are, usually, studied individually. In this thesis work the research focuses, in particular, on the first one, terrain traversability analysis.

Terrain traversability analysis aims, in practice, at allowing the rover to navigate in environments of varying complexity ensuring the achievement of the vehicle's mission's goals and its safety. It grants the UGV the avoidance of hazards that can potentially lead to damages to the vehicle or, in extreme cases, even its complete failure. In the same paper mentioned previously [20], other than a panoramic view

on the research in the field, referenced in the next paragraphs, a quite detailed definition of the term traversability is presented and is here reported to better understand the context and main components of the problem. Traversability is, there, defined as the capability of a ground vehicle to reside over a terrain region under an admissible state wherein it is capable of entering given its current state. This capability is quantified by taking into account a terrain model, the robotic vehicle model, the kinematic constraints of the vehicle and a set of criteria based on which the optimality of an admissible state can be assessed.
The mentioned terrain model includes perception information about the materials and terramechanic properties of the terrain, the interaction of the latter with the navigating vehicle and the geometric characteristics of the soil itself.
The approach to the traversability problem has been, firstly, considered as a binary problem, so only setting apart traversable and non traversable areas, while later on has been characterized as a continuous problem, being also able to differentiate between different degrees of traversability.
When discussing about the approach to actually evaluate terrain traversability, it is possible to divide them in several categories relative to the key concepts characterizing the methods. The first subdivision comes from the source of data gathered by the sensors in use on the UGV:

- Proprioceptive sensory data processing, using internal data of the robot (e.g. IMU, wheel slip sensors, collision sensors).
- Exteroceptive sensory data processing, taking advantage of data coming from the environment which surrounds the vehicle.

Proprioceptive data can be, of course, useful in the traversability analysis, but almost uniquely as refining information used to better an, already done, evaluation. This is due to the necessity of having already traversed, or being in the act of traversing, an area of terrain in order to retrieve information from this kind of sensors and, as it can be easily imagined, in order to ensure safety of the vehicle, the analysis must have been assessed before actually driving on that region.
For this reason literature focuses mostly on exteroceptive data and, among the different possibilities, two categories of approaches constitute the building blocks of the state of the art, **Appearance-based** approaches and **Geometry-based** approaches.

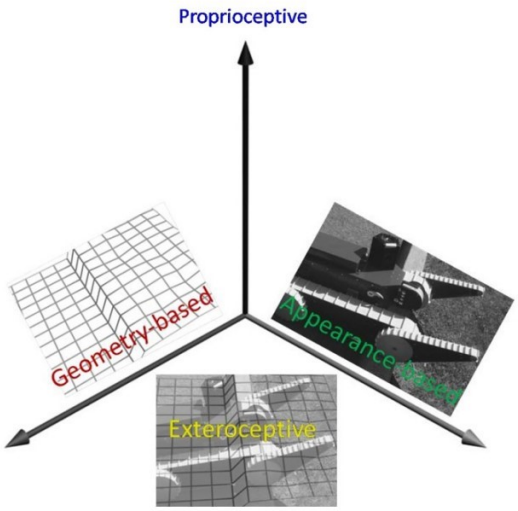

Figure 3.1: Approaches to terrain traversability analysis [20].

3.2 Appearance based approach

For what concerns the appearance based method, every operation in the realm of image processing, with either the aim of classification or regression of terrain characteristics, is included.
The possibilities are multiple and diverse, ranging from semantic segmentation of terrain images, to identification of the different classes of terrains the vehicle could encounter [21], to regression of terrain characteristics from visual information. Those data could range from RGB images [10] to other properties in the visual spectrum, like gray level [22], or even using multispectral images [23].

3.2.1 Semantic properties' analysis

Discussing about classification, the most important process, in the field of autonomous navigation, must be semantic segmentation, which was previously introduced in chapter 2.
The use of semantic information is fundamental in building, for the UGV, a terrain aware path which is informed of the various types of terrain surrounding it and which, among those, are the most dangerous for its navigation. The understanding of the different semantic classes, is central in order to avoid potentially hazardous situations which can verify even in geometrically safe terrains. A plain area, from a geometric point of view, can surely be safe in terms of obstacle avoidance, but could easily still be dangerous for the rover itself. To provide just a couple of examples, it could be a sandy region, which is well known to expose the vehicle to many risks, or a region with sharp rocks, even if really small, which could damage the wheels of the rover.

In semantic segmentation, the main actors are convolutional networks [21]; in this section it will be presented, with some additional details, one of the possible architectures which may be used to tackle this problem, UNet. It has been chosen to provide this more detailed description since it is the one employed, in this thesis work, to handle the appearance-based side of the approach.
UNet has been originally created, and presented in [24], with the aim of being employed on biomedical images, but its versatility allows it to perform at the same level in many different domains (e.g. [25]). UNet is a fully convolutional network, proposed in 2015, to improve the performance of what was the current state of the art, as an example, in the paper, is cited the network winner of the 2012 EM segmentation challenge at ISBI [26]. Among the drawbacks of the previous architectures fixed by the introduction of UNet, it is possible to find:

- Slowness of the strategy due to the need of running the network separately for every patch of the images considered, with a lot of redundancy when patches

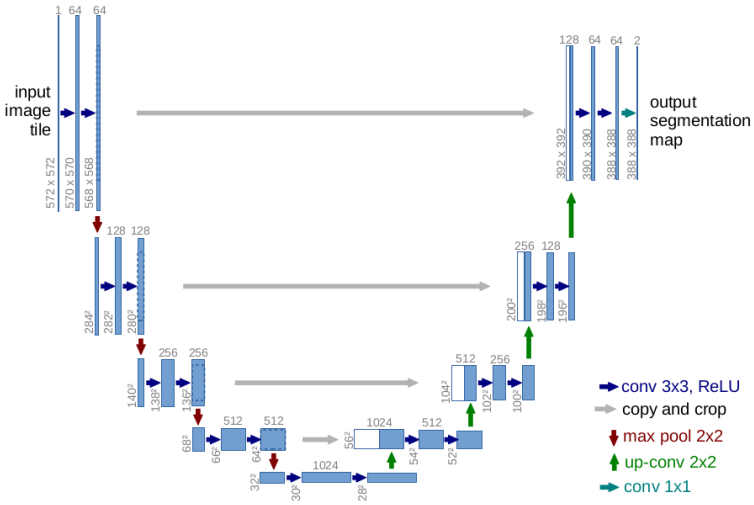

Figure 3.2: Architecture of UNet[24].

overlapped.

- Unavoidable trade-off between localization accuracy and contextual information. To improve the latter larger patches were needed, requiring more max-pooling layers which have a detrimental effect on accuracy and vice versa.

The new network, called in this way for its almost symmetric structure, is built upon the one presented in [27], but with a series of modifications which made possible to obtain a more precise segmentation even when working with very few images. The pooling operators, after the contracting path (the encoder), have been substituted by upsampling operators, the upsampled output is combined with high resolution features and then passed to a final convolution layer to build a more precise output.
In plain terms, the structure of the network is pretty usual with a contracting path and an expansive path, or decoder. At each depth level we have, on the left side, the repeated application of two 3x3 convolutions followed by a ReLU activation function with a downsampling operation executed by a 2x2 max pooling layer, which doubles the number of feature channels. On the right side instead, after an upsampling operation, there is a 2x2 convolution halving those feature channels, the concatenation of the feature map from the corresponding layer in the contracting path and two 3x3 convolutions, both activated by a ReLU. In the last layer of the decoder, it is possible to see in fig. 3.2 that it is also added a final 1x1 convolution before the output is created.
The connections between the two sides of the network are called skip connections, this particular way of using them is called concatenated skip connections [28]. It is

a way to enhance the performances of deep neural network by passing information from previous layer to subsequent ones. Many are the possible benefits of using those connections, other than just an improvement in the overall performances of the network, it can be obtained a more efficient training, better memory usage and a decreased risk of overfitting.

The structural innovations proposed with the introduction of UNet constitute a major change in what was the landscape of semantic segmentation and, from this basis, a series of other variations, with pros and cons, have been proposed. During the following years, until present days, further enhancements have been presented to try to improve even more what was achieved. To mention only a few, following the same encoder-decoder structure, with the aim of obtaining more sharp edges and try to decrease the number of pooling layer used, DeepLab architecture was proposed [29], to fix the speed problem of its predecessor FastFCN [30] was introduced and then, to increase even more the results obtained by it, a new version of DeepLab, DeepLabV3 [31] was presented.
A step aside from fully convolutional networks has been researched in recent years, represented by the employment of transformer networks for semantic segmentation. Transformer networks, are originally designed for natural language processing tasks, but have demonstrated significant progresses in modeling contextual relationships within images with the consequence of the investigation of transformer-based segmentation models [32] [33].

3.2.2 Terrain properties' analysis

Other than the identification, from a semantic perspective, of the terrain classes UGVs can face during their navigation, another crucial factor must be taken into account in this analysis, the presence of non-geometric hazards [34]. In this category can be acknowledged all the risks related to the physical properties of the terrain, slippage, roughness, sinkage, water content and many more. Several approaches presented in the research landscape have the characteristic of evaluating such parameters taking advantage of in-situ measurements, either using some kind of precomputed terramechanics model [35] [36], or even with the aid of additional scientific instrumentation [37]. These methods, even having high performances and accuracy, are effective only on traversed terrains being almost useless in case the rover faces a new terrain resulting, in those cases, ineffective in the detection of such hazards.
This possibility moves the focus on the research for a method which fixes it, so an approach able to adapt or to be, in some way, independent from previous knowledge. That's the reason why a number of other techniques proposed in literature are centered around vision systems and the estimation of terrain properties from images.

This problem, however, contrary to the extreme ease with which humans and some kinds of animals intuitively solve it, based on experience, it's far from being a trivial challenge for machines.
Among the different methods and extracted characteristics, some interesting ones focus on:

- **Slippage** and **sinkage** prediction from images, these parameters can be of great importance in avoiding, for examples very soft sandy terrains or slippery surfaces. In [34] an architecture able to compute semantic segmentation and also infer those mechanical characteristics of the terrain has been proposed. The approach in the prediction of the properties is linked with the output of the segmentation network, being the probabilities of the different terrain types for each pixel the weights used in the computation of the final value of the mechanical parameters associated to every semantic class. Some a priori information on the mechanical interactions between the wheels of the rover and the considered soil need, however, to be known to compute those values.

- **Water content** from terrain images, fundamental to avoid muddy and potentially hazardous soils. In both [22] and [38] the water content of terrain samples from images is estimated, but neither of these is centered on applying the method for traversability analysis purposes and further more to be applied in planetary exploration. The parameter is estimated taking advantage of surface gray level measurements in the first approach and values in red, green and blue channels of the images in the latter, but both of them require, again, some kind of prior knowledge. Respectively, soil surface gray level sensitivity and images of every type of terrain with specific percentages of moisture.

- **Roughness level** of each pixel in terrain images, of meaningful relevance in detecting areas where rover's wheels could have the right grip or face risks and even get damaged. The method is proposed in [10] where the parameter's value is estimated from only RGB images. The prior knowledge needed, in this case, is the availability of a dataset presenting roughness masks of the terrain images, but the authors also developed a method to retrieve it just from the knowledge of depth information, easily acquirable from many different sensors. This is the starting point chosen to be the basis of the approach followed in this thesis work, due to the more available reproducibility and the benefits that, such a system, could give to planetary exploration. As an illustrious example, it is possible to look, in fig 3.3, at the damages that NASA Curiosity wheels have reported traversing the rugged terrain of Mars surface, which could, hypothetically, be reduced including the knowledge of terrain's roughness in its traversability analysis.

Figure 3.3: One of the six aluminum Curiosity's wheels with visible gaps caused by navigating over rugged surfaces.

3.3 Geometry based approach

Most of the previous works on the topic of terrain traversability analysis are founded on a geometry based approach, so revolving around the capability of the UGV of estimating the geometric characteristics of the environment surrounding it and, from the gathered data, evaluating traversability of the terrain which the rover can, geometrically, navigate. This kind of method is, therefore, able to detect both positive obstacles, like rocks or bumps, and negative ones, like holes or harsh depressions, to understand where the terrain changes its slope and where, instead is possible to find more plane, and traversable, surfaces. The key concept in all the approaches in this category resides in the creation of a terrain model. From data gathered by sensors like lidars or stereo cameras, elevation maps can be built, according to which a series of features is derived. An elevation map (fig. 3.4), or digital elevation model (DEM) is a representation of the terrain surface, derived typically from range data, which describes the geometric conformation of the soil at different geographic positions. It can be obtained through a series of procedures starting with point cloud or depth map generation, depending on the available sensors, creation of a grid based representation and, in the end, a combination of interpolation and smoothing to fill possible gaps and reduce noise.

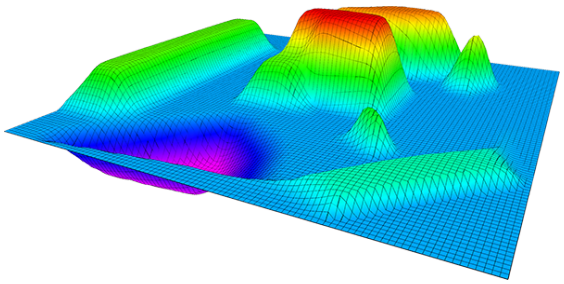

Figure 3.4: Example of an elevation map[39].

It is possible to divide the majority of geometry based methods into three groups depending on the way traversability features are obtained [20]:

- **Signal processing**, not as popular as the other two, focuses on the use of the elevation map as input is some kind of processing operation. In [40] for example, traversability features are extracted using Fourier analysis, while in [41] has been followed an approach based on wavelet decomposition.

- **Convolution with kernel**, the term convolution must not be taken literally in this case, but it is indicating the way in which the features are computed. In these methods the vehicle is approximated by a 2D kernel and the terrain map is iteratively processed with this window, centered each time at the total

set of positions and potential orientations of the vehicle. The rectangle is, in this case, representing the UGV and features are extracted only in the area interested by it.

- **Statistic processing**, has been the most popular and explored approach. Starting, again, from the data acquired by sensors, e.g. point clouds, elevation maps are built and then statistical analysis of the distributions of terrain features is used in order to determine traversability coefficients. In [42], for example, traversability of each grid cell is evaluated as the product of having permissible slope and roughness, in [43] is, instead derived globally as product of pre normalized measures of slope, roughness and obstacle presence. The cost of each grid cell was, then, assigned by local least-square plane fitting. A very advanced member of this category is also the method based from [44] and defined in [45]. This approach creates, as already seen previously, an elevation map from data of range sensors, but arranges it in a robot-centric fashion. The terrain map is associated with the pose of the robot and, at any time, constitutes a local representation of the surrounding terrain. The advantages of this approach reside in the focus on keeping always the highest level of accuracy closer to the robot, so in the most important area for the autonomous navigation of it, while decreasing the importance of what the rover has left behind.

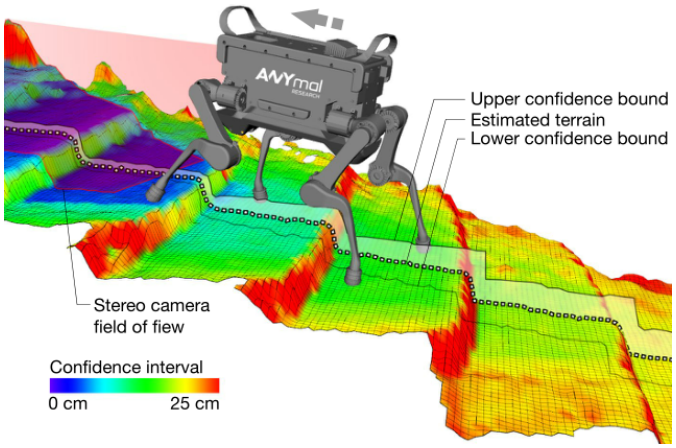

Figure 3.5: Local robot-centric map from the approach presented in [45].

3.4 Hybrid approach

After having reviewed the two main branches of terrain traversability analysis it is needed to point out some non-negligible flaws of both of them. If, for example, it is chosen to follow a pure appearance-based approach, it will be certainly possible to plan the rover's path being certain to give priority to more traversable classes of terrains, but would be impossible to judge the geometric conformation of the soil. Obstacle detection and avoidance would not be taken in consideration too, unless a specific class for every possible positive and negative obstacle is created. With these conditions, even the most traversable class of terrain possible could, in reality be extremely dangerous for the UGV opening the vehicle to risks which could, potentially, lead to disastrous consequences. Same thing can be said for an exclusively geometry-based approach. Considering the case of having an hypothetical perfect geometric evaluation of its surroundings, the rover will always choose, when possible, to traverse a flat surface, but without any knowledge of the materials of that surface. Considering the domain of interest of this thesis work, planetary exploration, the vehicle would have no mean to differentiate between a safe flat surface or a sandy terrain which, as mentioned in previous chapters, can be really hazardous for rovers.
These are the reasons which pushed research in the direction of hybrid approaches and determined the choice of one of them as the method employed in this work.
In some of the earlier researches on the theme, like [46] and [47], the approach was to use the geometric portion mainly for positive or negative obstacle detection while the appearance side to provide additional information. Respectively, the type of obstacle detected to see if it can be penetrated by the rover, like a bush, or not, like a rock, and to give a first, color based, classification of terrain types.
Another interesting procedure is the one presented in [48], where the two sources of data are combined together to be the input of a fully convolutional network classifying every pixel directly based on its traversability. The type of hybrid approach it has been decided to follow is, instead, pursuing a more balanced use of the two methods, both of them are employed in order to compensate each other imperfections combining the traversability cost maps they create into a final one, taking into account a wider range of features. In figure 3.6 it is possible to see the whole architecture of the method, presented in [49], with the different modules and how they are connected together. For what concerns the semantic portion, the obtained mask needs to be translated from the 2D pixel coordinates into 3D world coordinates and represented on a 2D robot-centric cost map. The costs of the different terrains' classes are assigned empirically based on how much they affect traversability for the specific application considered. For the geometric approach, instead, the previously mentioned strategy, presented in [45], is used to create the second cost map which will be, then, combined with the first one through a

weighted sum of the two traversability indices.

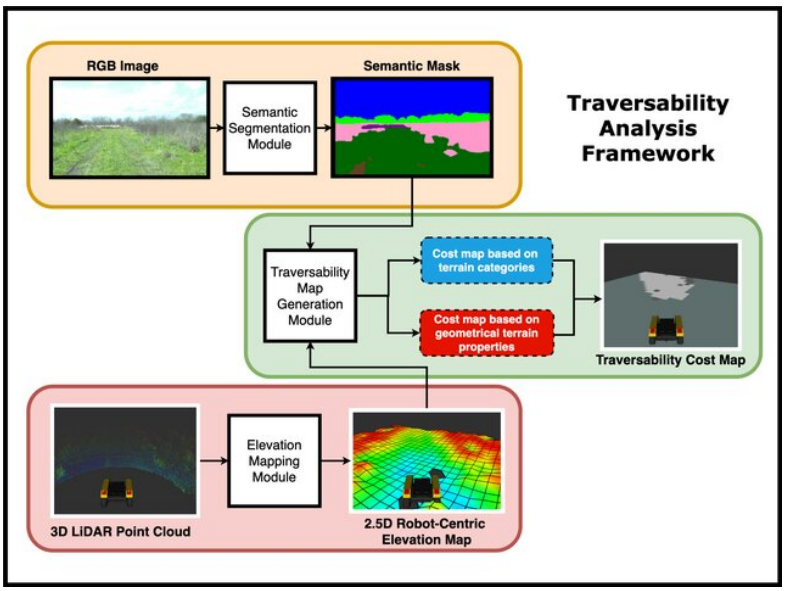

Figure 3.6: Complete architecture of the hybrid approach of [49].

In this work a similar approach to the latter is followed, readjusting the two methods to our choices and adding the visual evaluation of the physical properties of the terrain into the framework.
The structure described right above has been the main inspiration for the proposed framework, each of the methods has been re-adapted to match the ideas to be implemented and the limitations derived by the developing context:

- For what concerns the appearance-based section, as mentioned extensively, an additional module evaluating physical characteristics of the terrain in the acquired images is added.
- For the geometry-based approach a simpler strategy is followed, building a traversability map estimating the slope of the terrain through the information gathered from a point cloud. The initial project was to employ the method, by Fankhauser, mentioned in the last paragraph, but the plan was changed taking into consideration the significant complexity of the approach and the unavailability of the architecture in ROS2, which will be employed in the implementation of the framework.

Chapter 4

Methodology

In the following chapter a complete overview of all the components which characterize the proposed architecture is presented. After a first analysis of the datasets required to achieve terrain traversability analysis, a detailed description of the different modules employed to satisfy each task is provided and their combinations and interactions are reported.

4.1 Datasets

In the development of this thesis work one of the earliest, and most important, stages was the research and choice of an appropriate dataset. Being the studied approach focused on the domain of planetary exploration it was needed to find a set of images and, more importantly, of labels, coming from a reliable source and validated by scientific experts. To try to be aligned with the direction of current space exploration, the spotlight, as can be foresaw from the content of previous chapters, has been directed towards Mars and, thus, the necessity of obtaining a dataset of images and masks of its surface. When working with deep learning vision systems, like it is done for semantic segmentation and roughness classification, another requirement arises, the need of a large scale set. These same demands led to the creation of AI4Mars [50], the first large-scale dataset for training and validation of terrain classification models for Mars. It is particularly highlighted the size of this set due to the effect it can have on the performances of deep learning models. The dimension of the dataset plays a crucial role relatively to the learning capabilities of the network. A large and diverse dataset will enhance the generalization skills towards unseen data, improving the robustness of the training and having potential benefits also towards overfitting. A characteristic of deep neural networks, especially of convolutional ones, is the significantly high number of parameters, increasing even more the importance of generalization since,

with a higher number of samples, the model has more opportunities to adjust the parameters based on diverse examples. It is, though, needed to mention also that, even if the importance and benefits of a large scale set have been just stated, its size cannot be increased arbitrarily. In fact, it is needed to find the right balance between dataset size and other key factors. Mentioning just a couple of them, it is worth considering that as the number of samples increases also the computational resources needed to manage it increase and, moreover, the aforementioned benefits derived from the enlargement decrease their rate proportionally, to the point where it is more detrimental than beneficial to keep increasing its size.
Despite AI4Mars satisfying all the requirements for the semantic segmentation aspect of the architecture, being considerably large and coming from NASA JPL (Jet Propulsion Laboratory), so an extremely reliable source, it lacks a vital information for the whole appearance-based method, the roughness masks. This data, as mentioned previously, can be retrieved from the knowledge of depth masks which, again, are, currently, not available in the dataset.
For this reason, in order to train the complete appearance-based side, another dataset was needed. The images in this set are, however, synthetically generated through the use of a simulated environment and are coupled with both semantic masks and depth masks at pixel level.

4.1.1 AI4Mars

As mentioned in [50], the motivation behind the necessity of a large-scale- high-quality label dataset of Mars terrain images comes from the desire of obtaining terrain-aware mobility for planetary exploration rovers. Even if the autonomous navigation system of NASA rovers, AutoNav, has been employed, at various degrees, from the missions of Spirit and Opportunity it remains, also in its latest implementation on Perseverance, purely based on geometric information to assess traversability. This means that the teleoperated drives of those vehicles will comprehend also semantic evaluations made by human experts, while autonomous ones lack this capability. To try to fill this gap, researchers at JPL in [5] proposed a new machine learning-based terrain classifier for Mars named SPOC (Soil Property and Object Classification). It uses a deep convolutional network to identify terrain types and terrain features. The results obtained were encouraging, but to be able to obtain the necessary reliability levels for on-board algorithms standard a high quality dataset was needed.
The dataset includes the majority of existing high resolution images of Mars' surface. It gathers samples from several UGVs from MSL and MER missions taken with both NAVCAM and MASTCAM, two of the cameras in the equipment of the rovers. The set is split into four different label categories corresponding to the main types of terrain present on the red planet:

- **Soil**, one of the best terrain classes from the perspective of traversability. It comprises consolidated soil whose surface has sufficient cohesion such that significant slip conditions are not experienced by the rover. It can, often, present small gravel and sometimes also light wheel tracks left by the vehicle.
- **Bedrock**, hard surface of relatively flat and embedded rocks. It is usually drivable, but can, sometimes, be rugged. It includes all the surfaces in which height variations are under 30cm.
- **Sand**, similar to its terrestrial equivalent. It is covered with powdery and slippery dust making this class challenging to drive through, opening the vehicle to the risk of beaching. It is usually characterized by ripples and can also present deep wheel tracks left by the UGV.
- **Big rock**, the most non traversable, but also rare, class among these four. It contains all the rocks or rocky surfaces which height is bigger than 30cm, making it impossible for the rover to navigate over them.

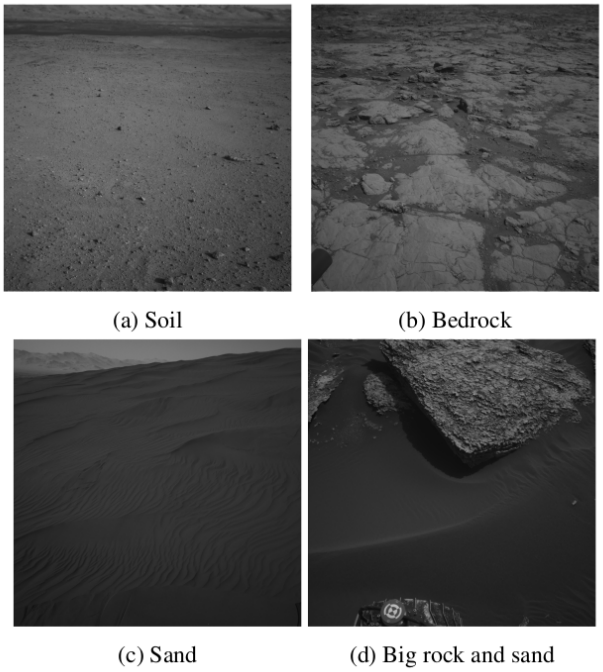

Figure 4.1: Representative examples of the four terrain classes [50].

Due to the remarkable volume of data to be labelled, the process was completed making use of volunteers' work through the platform Zooniverse. Citizen scientists

were administered a tutorial providing detailed information on each class and some illustrative examples. From a web interface for image annotation they were able to label images, randomly presented, until no more were available. Due to the volunteer nature of the process, to account for labelling inconsistency, many images were labeled several times and merged together to obtain a satisfying overall quality and reliability. For what concerns the test set, instead, to have a standard to compare results with, a smaller number of images, approximately 1% of the size of the training set, has been handled by experts. In order to have a balanced representation of all the classes, images were chosen sampling the principal different locations containing all the major terrain classes. It was, also, manually cleaned from low quality images, really similar ones and images in which the rover occupies significant regions of the picture. To each terrain class have been assigned three experts focused on the generation of high confidence labels and not on the maximization of the label coverage, implying that it is not uncommon to find, in this set, images with just small portions of terrain labeled.

As previously mentioned, AI4Mars is the first dataset which has been considered in this thesis work for the earlier development stages. Other than the four terrain classes, labeled with integers from 0 to 3, the set presents also pixels labeled with 255, used for sections which weren't assigned to any of the previous categories. For every image, two additional masks, other than the semantic segmentation one, are provided. They both contain additional information, employed to pre-process the images before creating the dataset object to be used in the scripts. The first ones, collected in the mxy directory, provide a mask of the rover, if present in the images, it can be used to signal the interested pixels and not consider them when operating semantic segmentation of the terrain. It has, instead, being applied to directly discard the images in which the rover explicitly appears, avoiding additional processing operations, given that it will not affect the learning capabilities being the dataset's size so large. The second kind of masks are range masks: during the creation of AI4Mars the labelers have been instructed to ignore, throughout the labeling process, features which where beyond 30m. These masks signal, in a binary manner, everything which falls out of the range limit and can be used to either mask out an entire area in which is not useful to operate semantic segmentation or, as did in this work, to merge those pixels with the ones out of the four classes. The ensemble of them has been casted into the new category of "No label" corresponding to the number four. In this way a more simple setup with just one semantic mask to take into account and an additional class, identifying unlabeled and out of range pixels, is obtained.

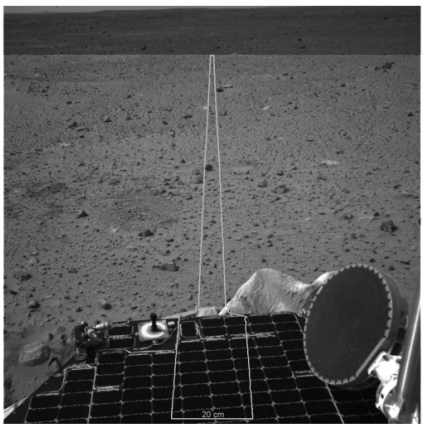

Figure 4.2: Darker area in the picture represents an example of the pixels cut off by the range mask (over 30m)[50].

As it is possible to see in figure 4.3, the big rock class is significantly less represented than the others. The main reasons behind this can be identified in big rocks being rare objects to find on Mars' surface and, even when encountered, occupying rather small portions of the images. It is common, in semantic segmentation, to have small objects as minority classes in the dataset, but it is a concerning event, in this particular case, being the object of this phenomenon the most dangerous and non traversable class of terrain. Being that much relevant, some kind of expedient needs to be employed to try and, at least, mitigate the impact of this class unbalance. Many are the possibilities, ranging from data augmentation of the samples in which the category is represented, to the choice of an appropriate loss function able to manage this phenomenon.

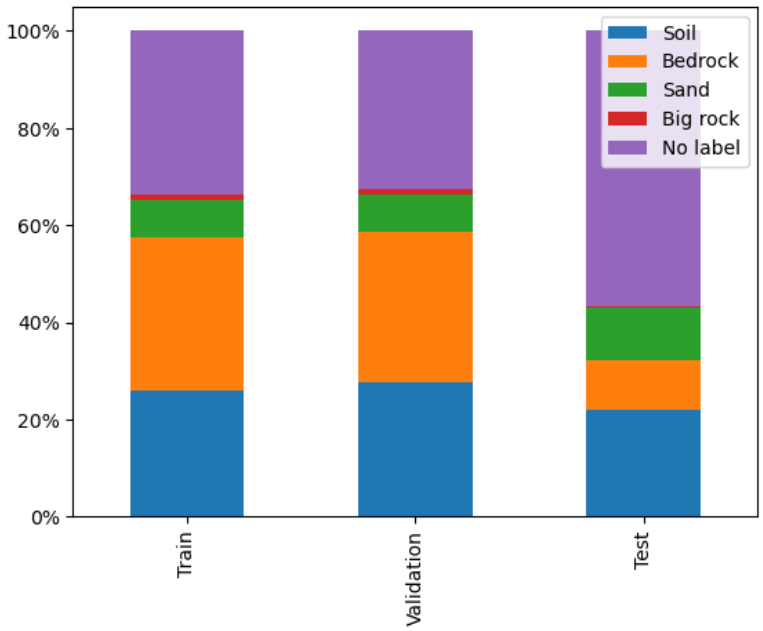

Figure 4.3: Percentages of the classes in an example of training/validation and test split.

4.1.2 Synthetic dataset

AI4Mars had on its side the large-scale and high quality of the labelling, but, as previously mentioned, lacked a portion of the information needed to fully train the appearance-based method. The first possibility which has been taken into account is the quest for a second, real, dataset coupled with depth data, necessary to build roughness ground truths. Unfortunately it was not possible to find an alternative which satisfied all the requirements needed and this conclusion led to the decision of creating a synthetic dataset, instead.
The generation of artificial images presents the advantages of having the opportunity to modify a lot of parameters, pose of the camera, texture used for each material, but more importantly to couple the RGB data with what was needed. In this way, it has been possible to obtain a series of samples correlated with semantic masks, with perfect precision, and relative depth mask, as required. To realize this, it has been used a simulation software called Oaisys, presented in 2021 in [51], based on the popular 3D computer graphics software tool set Blender.

4.1.3 Oaisys

Oaisys (Outdoor Artificial Intelligent SYstems Simulator), has been created to satisfy the open demand for high-quality synthetic data for planetary exploration tasks and to allow all the detailed operations required in the generation of those information. Among the most important features, provided by the software, it is possible to include the capability of a parametric development of the entire environment and the generation of high fidelity meta data. It is, in fact, possible to retrieve, for example, multilevel semantic masks or instance annotations, of paramount importance in many vision-based tasks in the field. The parametric nature of the set up is, moreover, capable of the creation of large amount of novel and diverse data, being able to use random deformations of the base surface, control the distribution of many objects and regulate atmospheric conditions like lighting and air dustiness.

Condensing the working principles of the simulator, a basic mesh, called stage, is created to be the ground surface, deforming it to obtain many different landscapes. On the stage, a series of materials is created using PBR (Physically Based Rendering) textures and is spread on the basic mesh. The textures can also be combined together randomly, using a noise shader, to create original materials which make the environment more natural. Every texture is assigned a specific semantic id making always possible to retrieve semantic information, even from the created combinations. On the generated terrain is possible to position objects, defined through blender based meshes and covered with appropriate textures, and scatter them around. For the scattering process a series of noise maps is available in the simulator to randomly spread them on the stage. After having defined terrain materials and objects, the simulator sets up the lighting of the scene either by selecting at random a sample among a series of HDR images, provided by the user, or by employing the sky simulator of Blender. The camera pose is decided either sequentially, from a *.csv* file, or randomly, inside intervals configured by hand which define position and angle of both the sensor and the invisible target, working as objective to point the sensor towards. It is possible to define how many terrain batches are rendered and how many samples to take from each batch. For every batch the values for the stage deformation, materials and object distributions are changed, while for every sample the camera angle and lighting conditions will vary. From every instance it is then possible to collect, other than the RGB image, all the necessary meta data, in our case the semantic mask and the depth mask.

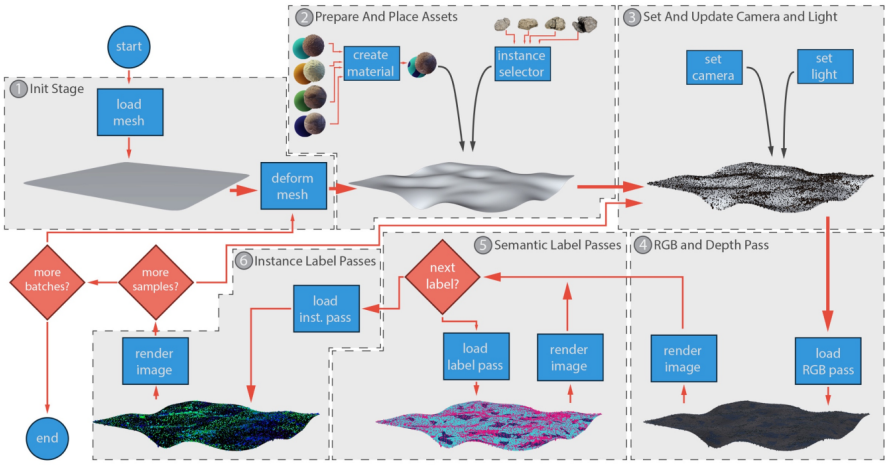

Figure 4.4: Example of functioning flow of Oaisys [51].

In order to obtain synthetic images as close as possible to AI4Mars ones, a couple of adjustments have been made before creating the dataset:

- Images' resolution has been defined equal to the real word samples, 1024 pixels by 1024.

- Sensor's position has been configured to be at the same height of the sensors used to create the images present in AI4Mars. The publicly available images are all from the MSL Mars mission of Curiosity, which, from the data presented in [52], has its NAVCAM at 1.99m and its MASTCAM at 1.97m. For this reason it has been configured to position the sensor at 2m with a random variation of 0.1m. To maintain the height constant, in the specification of the ranges of possible spawning positions of the camera the z value is kept at 0 while it is introduced variability in the x and y axis to try to increase diversity in the acquired images.

- Textures representing 2 terrain types of AI4Mars, soil and sand, have been selected and spread across the stage. For what concerns the bedrock class, instead, to obtain a similar effect to what we see in fig. 4.1, two ad-hoc blender objects have been created to reproduce groups of slabs of rocks placed over the terrain, and are positioned around in the simulation. The last semantic class, the big rock class, has been similarly obtained with other two types of objects, with different sizes and random rotations, scattered over the terrain. The sizes of the blender objects have been selected to satisfy the dimensions' constraints described in the AI4Mars dataset.

- The landscape preset "mounds", determining the overall shape of the environment, has been chosen, among the available ones, and to be closer to the martian landscape it has been tuned down to avoid having too many and too exaggerated elevations. An even more accurate work can be done by creating a configuration file from scratch, considering the morphological features of Mars to model, more accurately, the base mesh deformations and obtain an environment more resemblant of the red planet.

- To try to obtain a lighting condition more resemblant of the Martian surface, some of the parameters relative to the sun in the environment have been fixed to have its position always similar.

- Camera movements have been limited, allowing a bigger range on the horizontal axis, while accepting smaller motions on the vertical axis and on the proximity of the target to frame. This has been done with the purpose of trying to limit the depth of the images with the aim of having as much pixels as possible occupied by terrain. In this way, it has been tried to maximize the useful information retrieved from the samples maintaining a certain coherence with the AI4Mars dataset.

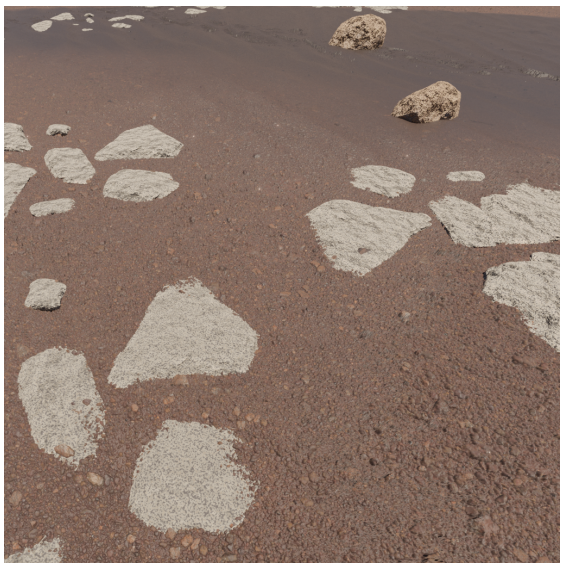

Figure 4.5: Sample of the created synthetic images in which all terrain classes are present.

4.2 Model structure

In this section it will be given an overall view of the structure of the complete model employed for this thesis work. After that, every section of the architecture will be detailed more to highlight the choices made and the motivation behind them. It is possible to see, in figure 4.6, a diagram which shows the full organization of the developed system.

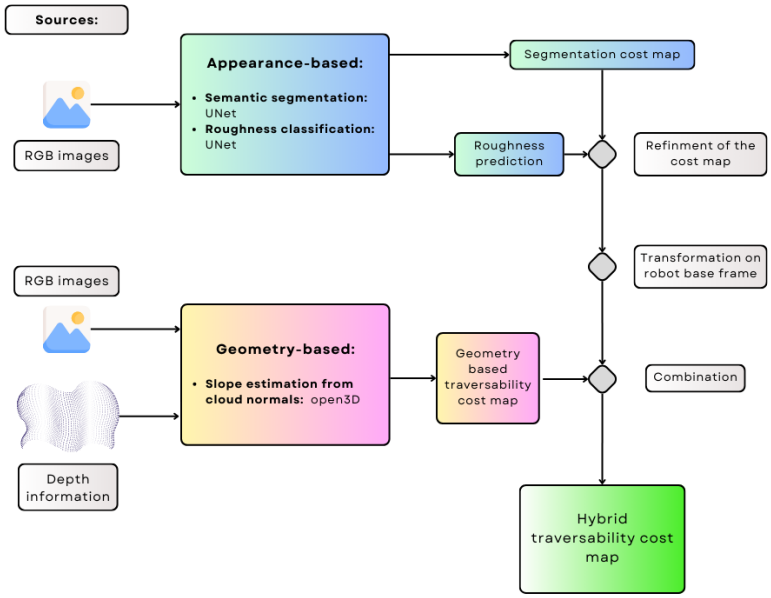

Figure 4.6: Complete model structure.

On the far left are positioned the two, exteroceptive, sources of data needed to complete all the operations, RGB images and depth data. They respectively come from a standard RGB camera and a depth sensor, either comprised in a RGBD camera or in a stereo camera, and are the starting point for the two methods which compose the hybrid approach. The appearance-based one encloses two sub-modules, one operating semantic segmentation and a second one, instead, in charge of classifying, from the same pictures, pixel-wise roughness levels. The cost map obtained from semantic information is, therefore, refined using the roughness mask to create the final traversability cost map of the appearance-based section. In the geometry side, instead, the point cloud data, obtained from the image and its relative depth information, are used to evaluate the normal vectors of each point and compute the angles between the latter and the vertical axis. From this information the slope of the terrain in each point is estimated assigning a traversability cost

relative to its value. To integrate the two sides together, to balance the lacks of each approach with the capabilities of the other, the appearance-based one is projected onto robot level and the produced output is, in the end, fused with the geometric one to obtain the hybrid traversability cost map to be fed to the path planner.

4.2.1 Semantic segmentation module

Dataset preprocessing

For what concerns the appearance-based approach, all the scripts have been coded in *python* language. The first thing done was the processing of the data into a dataset object. As anticipated, before the choice to switch to a synthetic dataset was made, semantic segmentation has been tested on AI4Mars. Four scripts have been implemented to perform segmentation: the one in which the dataset class is defined, the one in which the network class is created, the main script in which the training process is performed and a last one containing all the functions useful in the other ones. During the creation of the class *RealMarsDataset*, a preprocessing on the images was done generating the dataframes used as inputs by the class. In those operations, as mentioned previously, the images were read from their folders and their names, checking the relative rover mask, added in the training and test dataframes, only if the check gave a negative result. All the samples in which the vehicle appears explicitly have been, instead, discarded a priori to avoid additional complications. From the aforementioned dataframes, the paths of the directories containing images, segmentation masks, range masks and the transformations to apply to the samples, the datasets are created. The process has three main phases:

1. Image, mask and range mask are read from the repositories based on the names present in the dataframes, converted into *numpy* arrays and the appropriate data types.

2. The function *merge_mask* is applied to the segmentation and range masks. This process focuses on joining together the pixels labeled with 255 and the ones which, in the range mask, fall behind the 30m threshold. All of them are assigned to the same class, "No label", corresponding to number 4.

3. The last step is the application of the transformations to images and segmentation masks. All the samples are resized to smaller dimensions, 256 pixels by 256, to have lighter computations without losing too many details. Then, the *normalize* function from the *albumentations* library is used to ensure that the samples are correctly converted from the range of RGB images, 0 to 255, to the range 0-1. In the end, they are turned into tensors to be in the right format to be processed by the neural network.

Network implementation

The other main actor involved in semantic segmentation is the actual deep learning model. As discussed in previous chapters, the chosen one is UNet, a deep fully convolutional network. The implemented architecture is the same as the one displayed in figure 3.2 and has been implemented through *PyTorch*[53], a machine learning framework based on the *Torch* library. An instance of the class is defined based on its input channels, its output channels and the number of initial features to consider. The input channels, by default and in this specific case, are set to 3, being the red green and blue channels of the input images, the output channels are set to 5, which corresponds to the number of classes we are trying to classify each pixel between. The initial features, instead, are the output channels of the first double convolution block and, for the considered implementation, this parameter is set to 64. The network, summarizing its working principles, will take as inputs RGB images and yield as a result a tensor with the same dimensions of the original images. In correspondence to each image entry, the tensor contains the predicted probability that the pixel belongs to each class.

Training procedure

In the main script, where the training procedure takes place, as first thing, it can be found the definition of the setup. The learning rate has been chosen at a starting value of 10^{-3}, it is, then, controlled by a learning rate scheduler in charge of decreasing it gradually to avoid the possibility of having the loss curves being stuck in a plateau. To obtain a satisfactory decrease in the loss functions, the batch size has been selected to a quite small value, eight, given that UNet works better with smaller batch sizes, as discussed in [24]. The training, validation and test datasets, and relative dataloaders are then created from one of the functions in the aforementioned utilities script. This process, given the paths of the directories, the transformations to apply and the percentage of the training-validation split, generates the dataframes, uses them to obtain the dataset objects and, from those, creates the dataloaders. They are fundamental in the training procedure to allow the network the access to the elements of the dataset and to perform the subdivision in batches. It is worth mentioning that only the train loader will have its samples shuffled before each training epoch to provide higher generalization of the data. Once those data have been obtained, the frequencies of each class in the ground truths of the three sets are computed and are employed in the computation of the weights needed by the loss functions:

$$TrainWeights = \frac{1}{FreqTrain} \cdot \frac{1}{numClasses} \tag{4.1}$$

$$ValWeights = \frac{1}{FreqVal} \cdot \frac{1}{numClasses}$$

These weights are necessary, as mentioned earlier, to address a problem present in section 4.1.1 regarding the unbalanced nature of the dataset. To handle the low representation percentage of the big rock class, among the different listed possibilities, it has been chosen to select an appropriate kind of loss function, **weighted cross entropy** (eq.4.2). This function is exactly like the standard, multi-class, cross entropy but with the possibility to assign a weight to each class. These weights have the aim to increase more the contribution of specific classes to the value of the loss, focusing more, as a consequence, the learning process on these classes.

$$CE_W = -\sum_{i=1}^{n} \sum_{c=1}^{numClasses} w_c \cdot y_{i,c} \cdot \ln\left(p_{i,c}\right) \tag{4.2}$$

The automatically computed weights can, if needed, be refined manually to fine tune the learning process more and try to obtain the best performances possible. For these reason a specific function has been created allowing the possibility to easily tune more the weights either by reducing or increasing their magnitude.

The successive section of the script presents the definition of the training function, the process which encloses all the operation regarding the learning phase which are executed during each epoch. Here, batch by batch, the data, inputs and ground truths, are extracted from the training loader and fed to the neural network to obtain predictions for the semantic segmentation. The latter are then used, together with gts, to compute the value of the loss function for that particular batch. These operations are all executed employing the *automatic mixed precision package* of *PyTorch* library, which allows the simultaneous use of both standard float datatype and lower precision floating point, when feasible, with the aim of speeding up computations as much as possible. With the value of the loss, computed in the previous step, the optimizer is updated in the process called backpropagation. This is the technique in which resides the core of the training procedure, it propagates the value of the error, evaluated through the loss function, from the output layer to the input layer of the network, updating the weights of each of them based on the gradient of the error with respect to the weights. The objective of this procedure is the minimization of this difference, which materializes in the maximization of the learning capabilities of the model.

The next two parts of the script are the ones containing the main portion of the code. In the first one all the actors involved in the training process are set up, the model and the training and validation loss functions are created. The optimizer is initialized, in this case as an instance of the Adam optimizer, a popular optimization algorithm presented in [54]. It is a stochastic gradient descent method based on adaptive estimation of first-order and second-order moments. The advantages, of using such an optimizer, can be found in its capabilities, highlighted by its creators, being it computationally efficient, with little memory requirement, invariant to diagonal rescaling of gradients, and well suited for problems that are large in terms of data/parameters. Another important component which is initialized in this phase is the learning rate scheduler, in particular the *step learning rate scheduler* provided by *PyTorch*. Its function is to gradually modify the learning rate by a specific amount decided by the user, in this case decreasing it by the 30% every 2 epochs. Adjusting dynamically the learning rate a more efficient exploration of the loss landscape is possible, avoiding situations like plateaus, leading to better training outcomes.

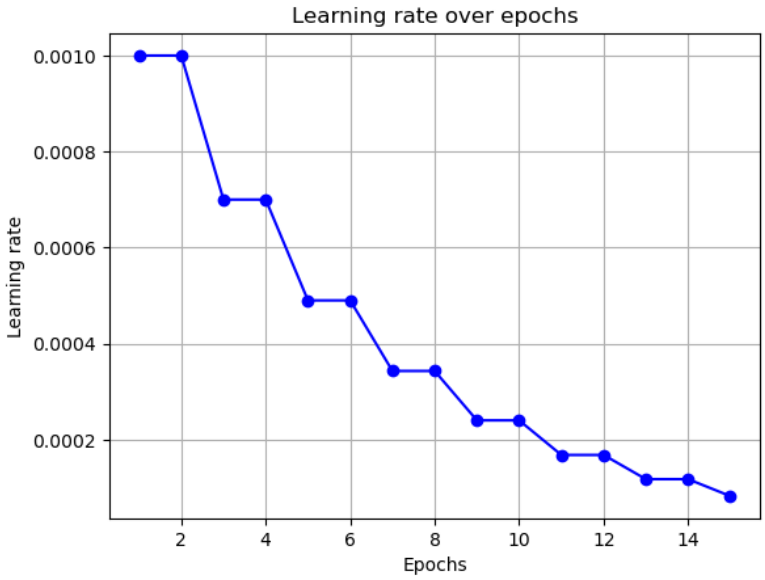

Figure 4.7: Learning rate decay using a step learning rate scheduler.

The last important components defined in this section are the evaluation metrics, fundamental in the assessment of the performances of the model; a detailed description of them and the motivations behind their employment will be presented in section 4.3.

The second main part of the script is the portion containing the actual training loop. Inside of it, during every epoch, the training function is re-executed, computing the values of the loss for the training dataset and updating the weights through the optimizer. The current state of the latter and of the network are then saved in a checkpoint, in order to be able to recover from where the last session ended, either to do additional training epochs or just to be able to reload the state into the model to have available the trained result. In the end, the model is set into evaluation mode to perform metrics evaluation on both training and validation samples and the cycle will start back from the beginning of the successive iteration.

Shift to synthetic dataset

The next step, as comprehensively discussed previously, has been the decision to permanently shift from real to synthetic images. To have a unique formulation, which could be employed both in semantic segmentation and roughness classification, avoiding the necessity of the creation of two separated sets, some adjustments to the previous dataset class were needed. The most important detail to highlight is the need to add, to the segmentation mask, the information relative to the roughness ground truth. In this way it has been possible to use the same dataset for both operations, just switching which channel of the mask we consider when retrieving the ground truth. Another fundamental requirement has been the deletion of the fifth class *No label*. This has become necessary as a consequence of the conditions imposed in the creation of the dataset itself. To have image samples effective for both semantic segmentation and roughness classification, the camera has been oriented in such a way to include as much terrain data as possible in a certain distance range, avoiding to collect information relative to areas so far that are not useful to analyze. This resulted in the creation of a set lacking even a pixel dedicated to the sky, which was the only area labeled with that class without employing a range mask as done in AI4Mars. Being it not represented, it results in a division by zero in the weight computation and, inevitably, in some obstacles with the evaluation of the metrics. To avoid any sort of problem the easiest and most practical modification to perform is to directly remove the class and train the network over four categories instead of five. The only criticality which could be faced as a consequence of this choice arises in the possibility of the method's validation with the real images. Being the presence of that class consistent throughout AI4Mars it could cause some problems in segmentation of images in which the sky appears, unable to be recognized by the synthetically trained model. The problem is though mostly visual since, for what concerns traversability the performances of the network for completely inaccessible areas are irrelevant. In order to obtain easily understandable samples to compare the

performances of the network, this aesthetic flaw would be corrected through the use of the provided range masks. Applying them to the predictions it would be possible to cut off completely the area interested by this potential problem and compare only the relevant data of the ground truth and prediction. The same process could be possibly used also for completely new samples in the presence of depth information which could substitute the range masks in the task of isolating only a region of interest which excludes the sky.

For what concerns the semantic segmentation scripts for the synthetic set, apart from the aforementioned discrepancies, everything has been coded in the same way to have the obtained results as comparable as possible.

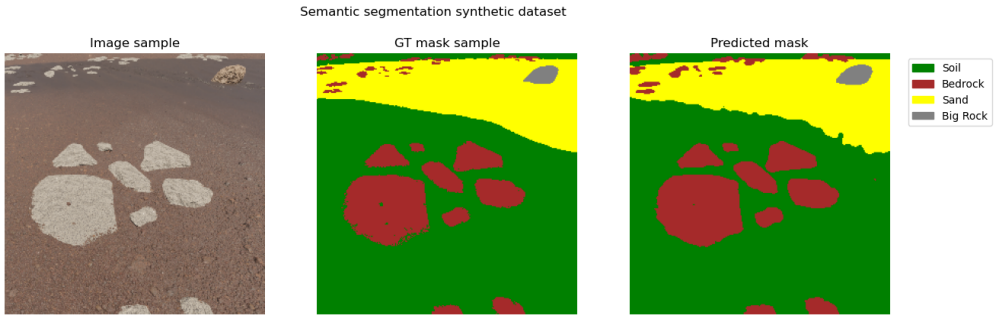

Figure 4.8: Sample of semantic segmentation performed on one of the images of the test set.

Semantic segmentation cost map

The last piece missing from the puzzle is the creation of the traversability cost map relative to semantic information obtained from the predictions of the network. Being able to retrieve the data regarding the different classes of terrain surrounding the vehicle, it has been necessary to assess a method to translate them into a traversability cost. The way in which the function creating this cost map has been designed is centered around two main choices:

- **A priori assumptions**, in order to assign a cost to a certain terrain type some kind of evaluations need to be made. In this particular case the costs must been empirically decided based on both the experiences of prior planetary exploration missions and the specific structural characteristics of the vehicle applying the framework. For the case of rovers navigating on Martian soil, it has been experienced that the more dangerous classes are sand, in which as mentioned earlier some vehicles found their end, and the untraversable big rocks.

- **Cautious approach**, in order to make the evaluations "uncertainty-aware" the cost map is designed to take into account also the confidence of the predictions of each pixel in a specific class.

These two concepts have been integrated together in the creation of the cost map. Instead of assigning a cost to each class, a cost reduction has been coupled with every category. This reduction is then multiplied by the confidence of the prediction allowing the result to be lower when the classification is not so certain. In this way, even among the same class, if some areas have been predicted with lower confidence the resulting cost assigned will be higher, increasing as possible the safety in the choices the UGV will make.

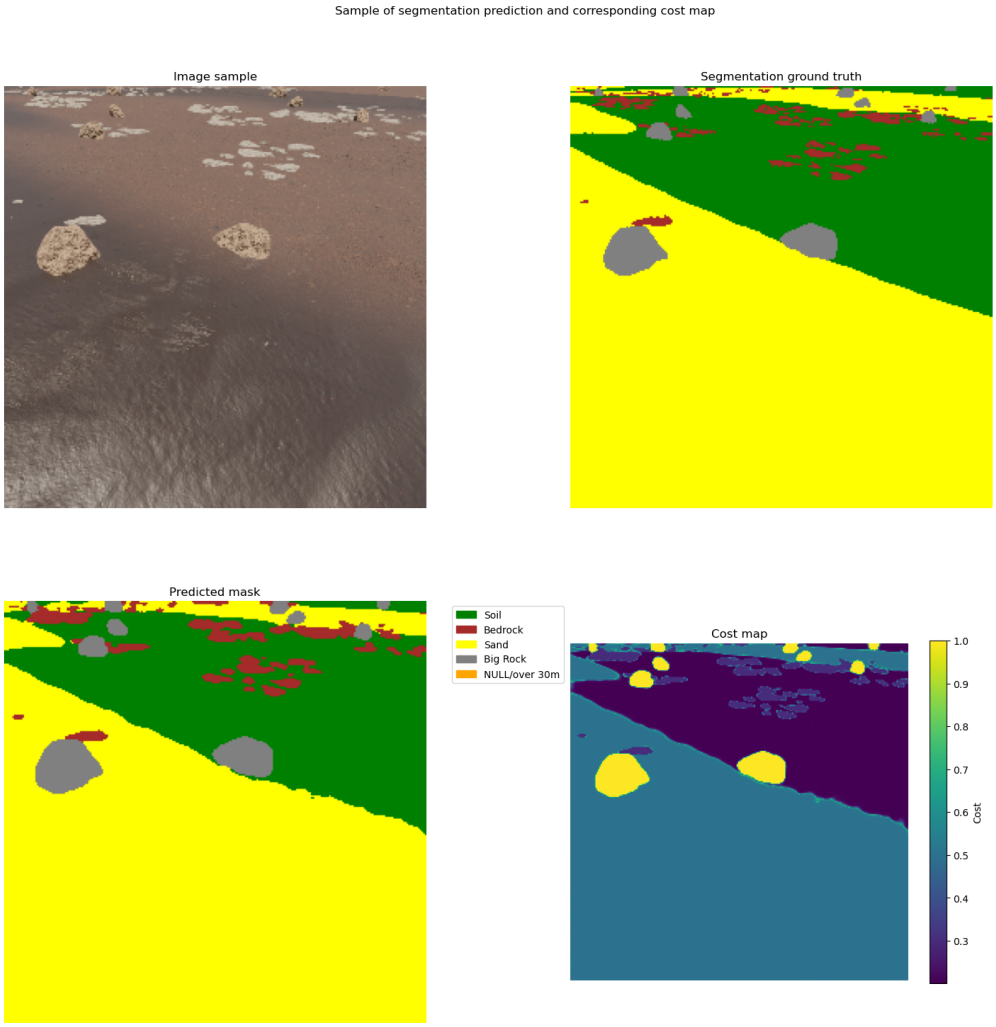

Figure 4.9: Sample of cost map based on semantic segmentation.

It can be clearly seen in the figure above how the cost assignment follows the described guidelines, each class has a different cost, with the higher values to sand and big rocks. Moreover, mostly at the borders between different terrain types, it is also possible to notice a higher cost, consequence of the lower confidence in the prediction.

4.2.2 Roughness classification module

The next component of the appearance-based section is the roughness module. The first idea for the realization of such component has been to perform a pixel level regression of roughness values of the terrain in the gathered image samples. This task goes beyond the ways in which, normally, this parameter is estimated making no use of sensors evaluating the geometric characteristics of the soil. As for the semantic segmentation, the key actor of the module is a neural network. In [10] it is presented with the objective of regression instead of classification, meaning that the output of the network will be a mask with the same dimensions of the RGB image containing, for every pixel, the predicted roughness value. As will be further explained in the next paragraphs, requiring an evaluation of the predictions' uncertainty to build the cost map and being the resulting data used only as a refining measure over the semantic segmentation one, for what concerns the task in exam, the exact determination of the roughness value of each pixel is not necessary. For this reason it has been, ultimately, decided to resort to roughness classification.

Roughness ground truths

The starting point for the realization of the module has been the creation of the roughness ground truths. As mentioned previously, the information obtained from the simulator are limited to the depth maps retrieved for every image, the next step is the creation, from the available data, of equivalent roughness maps. The method presented in [10] has been followed with some adjustments which will be highlighted.
It can be summarized into four steps:

1. **Point cloud creation**, here is introduced the first deviation from the paper. Differently from the approach described in the aforementioned research, the cloud is obtained from the *python* library *Open3D*, proposed in [55]. It is created starting from both the depth map and RGB image, to use as much data as possible. The final result is achieved in two steps, at first an RGBD image is created through the function *create_from_color_and_depth* and it is, then, used to create the point cloud taking advantage of the method *create_point_cloud_from_rgbd_image*.

2. **Plane fitting**, the method proposed in the paper evaluates plane fitting on a dataset of images constituted by close ups of terrain surfaces which, in 3D, correspond approximately to the size of the wheel of the robot employed for the testing. Being the images of the synthetic set really different, having decided to use them for both appearance-based modules, it has been limited the area represented in the samples controlling the way the pictures are taken. Another adjustment made, differently from the paper, is the application of the plane fitting algorithm on limited sections of the image in an iterative process until all the pixels have been covered. It has been decided to follow this approach to try and obtain a roughness mask as precise as possible to be sure that the surface irregularities are all highlighted in the ground truths. In the utilized method the RGB image is divided into a series of patches, which dimension is modified depending on the depth values. After deciding the extremes of the patch size, selected at 8 and 30 pixels after a couple of experiments, its value, starting from the lowest for furthest points, increases as closer ones are considered. The aim of this process is to have always the size allowing to capture as much roughness variations as possible. For each patch, a corresponding point cloud is created and plane fitting is applied, trying to find the coefficients of the plane that best approximates the studied surface. The set up of the fitting procedure is slightly different from the one which can be found in the paper, since open3d uses a different algorithm, RANSAC instead of K-nearest-neighbour. It is also possible to tune three important parameters in the used *segment_plane* method:

    - ***distance_threshold***, maximum distance from the plane a point can have to be still considered an inlier. It has been imposed to a value of 0.01 given that it grants a satisfactory level of accuracy.
    - ***ransac_n***, number of initial points to be considered in the first iteration to fit the initial plane. It has been chosen to a value of 9, which is a reasonable number given the dimensions of the patches considered and the quantity of possible noise in data points of each patch. The minimum possible value would have been 3, being it the minimum number of points to define a plane.
    - ***num_iterations***, number of times the algorithm is reiterated to reach the final result. It has been selected to 10000 to be sure to have, again, a good accuracy level.

3. **Roughness computation**, once the plane coefficients for a patch have been obtained, the following formula for the computation of the roughness level of each point, presented in the paper [10], is employed to retrieve the value.

$$r_i = \frac{|-d - a \cdot x_i - b \cdot y_i - c \cdot z_i|}{\sqrt{a^2 + b^2 + c^2}} \tag{4.3}$$

4. **Roughness map definition**, the roughness value computed in the previous step, is assigned to the relative pixel in the image making use of the correspondence between the point's index, in the unstructured point cloud, retrieved by the *points* method, and the index of the pixel in the 2D numpy array defining the map. A post-processing operation is done on the obtained map cutting the roughness values at mm scale, imposing every lower roughness value to be equal to 1mm. This operation is done in order to keep only the relevant data which are useful to consider in the regression process. The resulting array is then saved and will constitute the roughness ground truth of every sample in the two datasets.

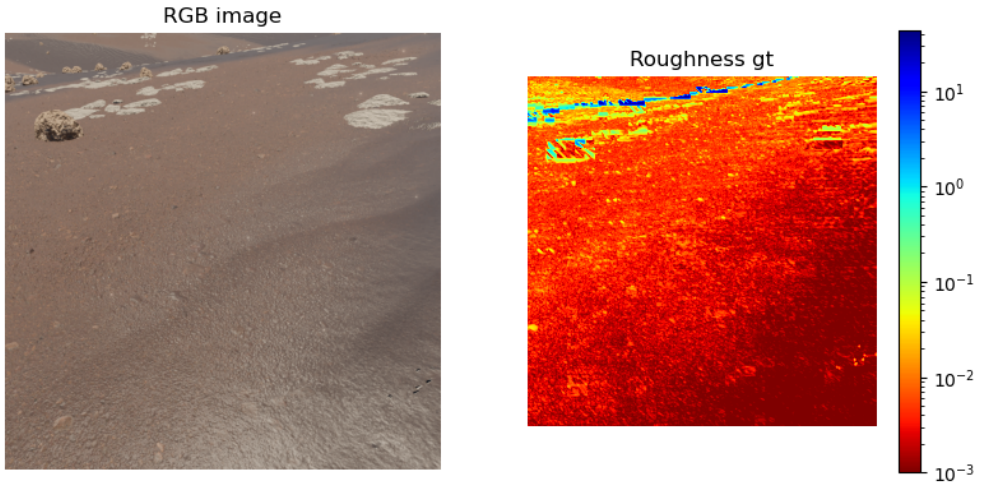

Figure 4.10: Sample of RGB image and relative roughness ground truth measured in m_s.

Dataset creation

The next step, after the creation of the roughness maps has been completed, is to perform the adjustments to the dataset class, mentioned in paragraph 4.2.1. To make it contain also the newly obtained data, an additional layer to the segmentation mask has been added.

Architecture of the module

The original architecture of the model used in [10] (summarized in figure 4.11), translated from its presented implementation in Caffe [56] to PyTorch terms, is composed of six layers, respectively:

1. **First 3x3 convolutional layer**, a standard convolutional layer with a 3x3 kernel followed by the application of batch normalization and activated by a ReLU.

2. **First up-projection layer**, first of the two belonging to this category. The up-projection type needs to be defined as a new class since it is the result of the combination between different layers. In particular, its input is firstly processed by an upsampling layer with scale factor equal to 2, then the output of this step is fed both to a 5x5 convolutional layer, followed again by batch normalization and ReLU activation, and to a simple 3x3 convolutional layer. The outputs of the two branches are handled differently, the first one is processed by a following 3x3 convolutional layer with batch normalization and its final result will be elementwise summed with the second output. The last operation is a ReLU activation applied to the previously computed sum.

3. **1x1 convolutional layer**, a standard convolutional layer with a 1x1 kernel followed by the application of batch normalization and activated by a ReLU.

4. **Second up-projection layer**, a second instance of the up-projection class.

5. **Second 3x3 convolutional layer**, same structure as the first one.

6. **Final convolutional layer**, last convolution applied to the processed data, followed by a ReLU activation which yields the final output of the module.

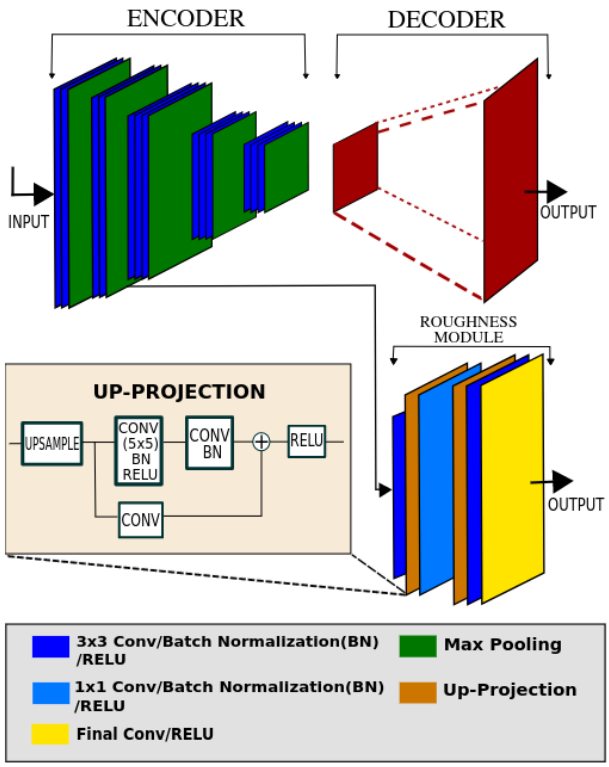

Figure 4.11: Architecture of roughness regression module[10].

The segmentation network in the regression task, as described in the paper, it is used in order to take advantage of its feature extraction capabilities to derive the relevant features which constitute the input of the module. It serves as a kind of encoder working as a preprocessing filter, yielding the data in a form which can be directly the input of the module. This means that the interested section of the segmentation network, from the start of the encoder to the second pooling layer, is copied and turned into a separate class defined in the script *Encoder_filter*. It will be, therefore, preloaded with the final checkpoint, saved after the end of the segmentation training, to gain the aforementioned feature extraction capabilities.

Cost map incident

As previously mentioned, the aim of the whole regression process is, as for the semantic segmentation, the creation of a relative cost map. To complete this task, it has been decided to follow a similar approach to what done for the segmentation map, assigning a traversability cost based on the roughness level, re-normalizing the values in the range 0 - 1, and then multiplying the obtained preliminary cost map

by the predictions' confidence. This process would allow the cost to be balanced by the uncertainty in the predictions, to have a more cautious assignment. The necessity of deriving in some way the confidence of the regressed values made arise a problem: a standard regression network does not return in any way a measure of the uncertainty for its predictions. The state of the art, for what concerns uncertainty prediction, is represented by the use of Bayesian Neural Networks[57]. They introduce stochasticity in the network's architecture allowing to retrieve the mean, to be considered as point estimate, and the variance, representing instead the uncertainty, of the predictions.
Being, for the purpose of this research, more important to have a measure of the latter than to have a continuous regressed roughness estimation, classification has been preferred. These are the main reasons which motivated, as anticipated in the very beginning of section 4.2.2, the decision to operate roughness classification instead of regression.

Roughness classification

The accuracy level achieved by classifying roughness into several classes is more than enough to be able to refine the result obtained from semantic segmentation and, moreover, as already seen for the previous appearance-based task, this kind of networks are capable of yielding a kind of uncertainty measure of their predictions, returning, for every pixel, the probability it belongs to each of the classes it is trying to differentiate among.
To perform roughness classification the first adjustment needed revolves around the dataset creation, a classification ground truth is required to train the network. To obtain such data, a series of roughness classes needs to be defined a priori. The decision process is governed by both a trial and error procedure and a task-related choice, linked to the environment depicted in the images and the navigation capabilities of the rover. It has been decided to keep the number of classes limited, trying to avoid increasing the complexity of the problem past a required level and, for this reason, four roughness levels have been defined:

1. **LV0**, or *"Negligible roughness"*, between 1 and 4 mm. This level is not representing in any way a danger to the traversability capabilities of the vehicle being the surface very smooth.

2. **LV1**, or *"Low roughness"*, between 4mm and 3cm. Is reasonable to think that this class starts to have some kind of effect on the navigation of the UGV, but even the higher end should be easy enough to be handled.

3. **LV2**, or *"Medium roughness"*, between 3 and 10 cm. This class has a relevant impact on the vehicle which, on the higher values, will face a significantly rugged terrain or even some examples of positive obstacles.

4. **LV3**, or *"High roughness"*, above 10cm. The last level represents the most hazardous roughnesses for the rover, it may be either really hard and dangerous or even impossible to traverse this class. It is found usually on the borders of instances of the big rock class or at the edges of some drastic changes of terrain slope, like steep hills.

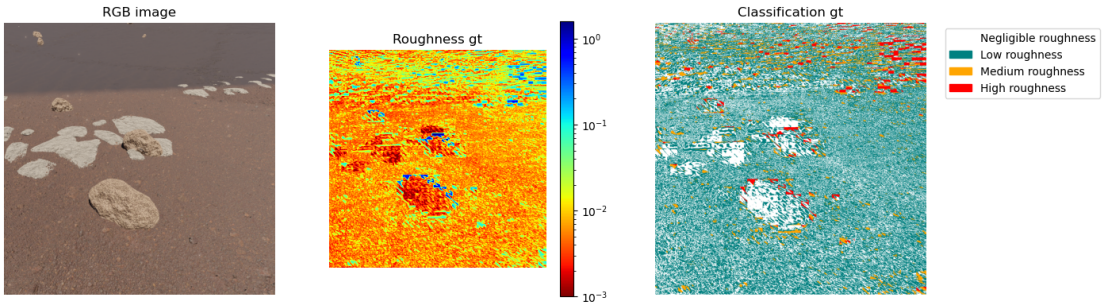

Figure 4.12: Example of classification ground truth.

Once that the classes are defined, starting from the ground truth obtained from the previously seen method, each pixel is assigned to a roughness level in the dataset creation, obtaining the classification mask. The network too has to be changed resorting again to the same UNet implementation used for semantic segmentation, re-adapting also the training scripts to make them suitable for the new task.
The strategy employed in the training process has been the same, using again a weighted cross-entropy loss, due to the significantly unbalanced dataset obtained, and a step learning rate scheduler to gradually decrease the learning rate as the procedure goes on.

Training procedure

The training script is similar to the one described for the semantic segmentation network, with a few changes due to the slight differences of this task. The first step has been the definition of the hyperparameters' setup of the procedure. Due to the similarities of the two processes, the chosen values are the same as seen previously in the classification of terrain classes. The training, validation and test datasets and dataloaders are, then, created using the same functions employed also for segmentation purposes. The main difference, can be found in the selection of the ground truths which, for this task, are positioned as an additional layer of the previously created masks. Afterwards, the loss function is defined, right before the definition of the training function. The chosen criteria is again the same weighted cross-entropy discussed in previous sections, to be able to cope with the problem of classes unbalance in dataset creation. The training function, contains all the

operations done each iteration of the procedure, asks for the same inputs as for the segmentation case, but uses a different ground truth in the loss evaluation.
The first main section of the code is the one in which all the actors involved in the training loop are set up. Here, the network instance is created, the training and validation loss functions are initialized, followed by the Adam optimizer. At last, for both training and validation set, all the metrics employed also for semantic segmentation are initialized, to try to have the most complete view over the performances of the network during the training procedure.

The second main section of the script is the one containing the training loop. Here, during each epoch, the train function is called, to compute predictions, evaluate loss values and update the weights accordingly. Then, a checkpoint is saved, to be able to restore the training procedure from where it ended and to use the trained network without repeating every time the process. In the end, the metrics and loss values for both training and validation sets are stored in the relative arrays, to keep track of the progresses. The model, from the evaluation mode needed to compute the metrics, is restored to training mode and the cycle starts back from the beginning.

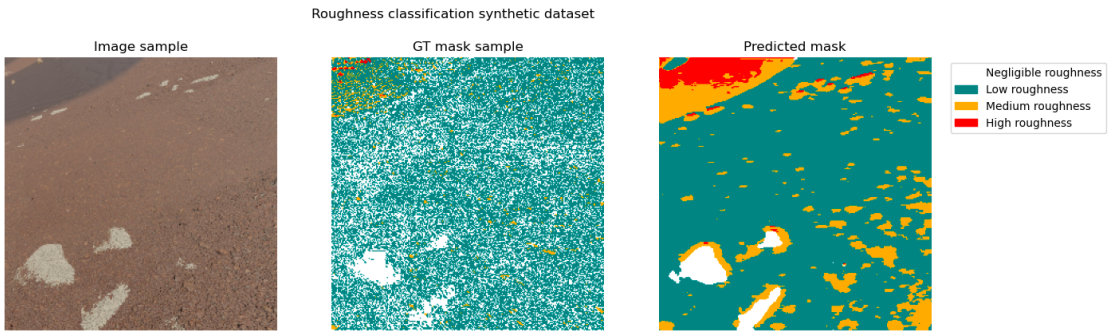

Figure 4.13: Sample of roughness module's output.

Roughness classification cost map

As per the semantic segmentation module, the final objective to achieve is the creation of a traversability cost map. In this case the map will be based on the roughness of the terrain surrounding the rover, having a higher value for rougher areas and a lower value for smoother portions of the soil. It has been previously mentioned that this evaluation is needed to refine the result already obtained from the segmentation network. The aim is, not only, to have a better understanding of the soil the vehicle will traverse, but mostly to increase the level of detail in the cost assignment to sections belonging to the same terrain class. The strategy

adopted is exactly the same detailed in section 4.2.1, the only difference resides in the considerations made for cost assignment. In this case the cost value depends on the increasing roughness and, again, takes into account the uncertainty in the classification as a form of additional safety measure.

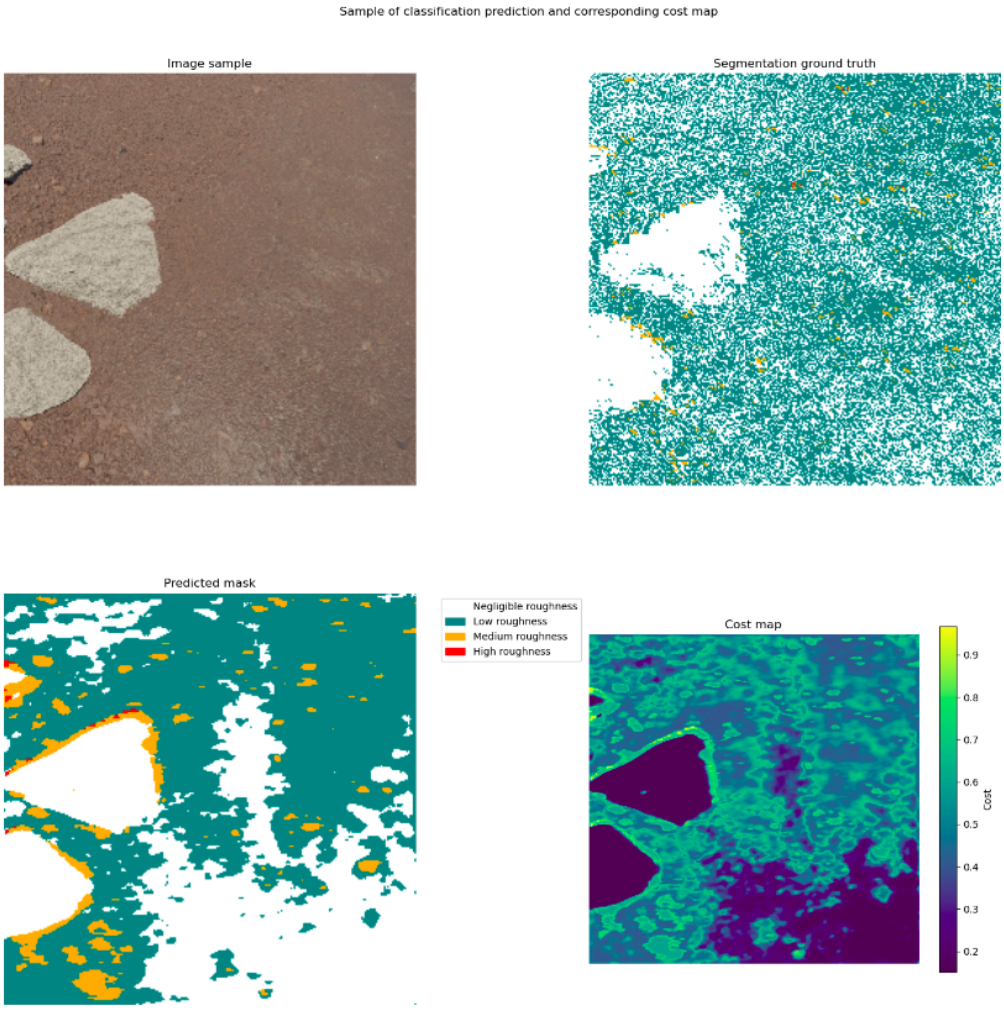

Figure 4.14: Sample of roughness classification cost map.

Appearance based cost map

Once the two cost maps, coming from the different appearance-based modules, have been elaborated it is needed to combine them together. This task is completed through the use of a weighted sum of the maps, a widely employed approach, assigning weights proportionally to the importance that, empirically, is desired to

assign to each of them. In this particular case, as extensively mentioned previously, the roughness analysis is considered as a refining measure to have a more complete and accurate comprehension of the terrain classified in each semantic category. For this reason, the weight corresponding to this cost map will be smaller than the one chosen for the other one. After some testings, the most appropriate values have been set to be 0.7 for the semantic segmentation cost map and 0.3 for the roughness classification one. This weighting, strategy grants an increased detail level in the cost assignment, as desired, without giving too much importance to the roughness classification which could, in some cases, have a detrimental effect given the overall lower performances due to the higher degree of the task's complexity.

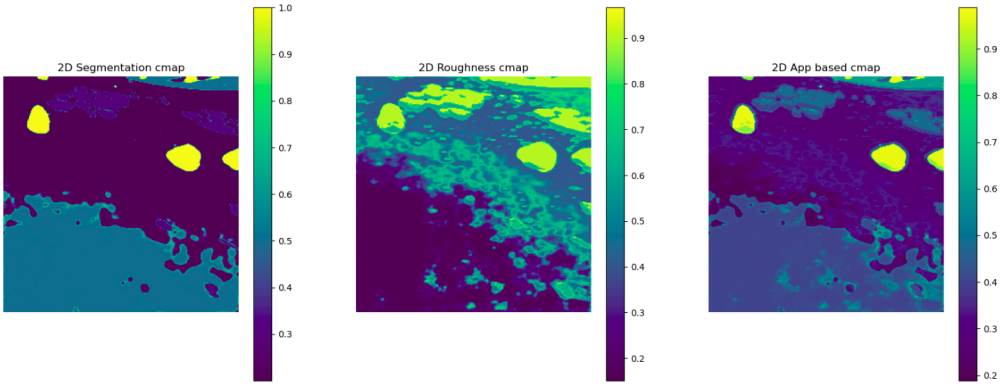

Figure 4.15: Combination process of the cost maps.

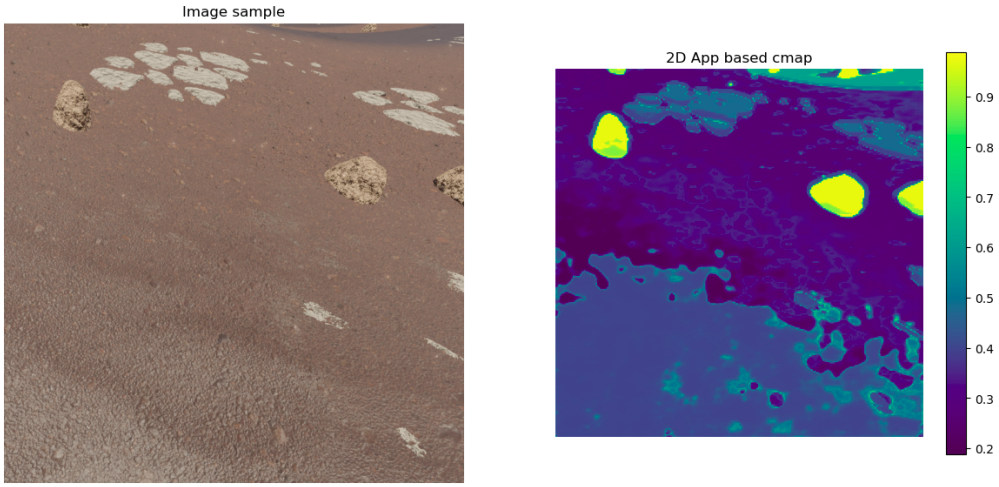

Figure 4.16: Sample of appearance based final cost map.

The weights' values are always chosen to sum up to 1 in order to be able to have a

result which still is included in the range 0 to 1.

4.2.3 Geometry based module

In section 3.4, the geometry-based side of the approach, intended to be followed, in this thesis work was introduced. As already mentioned, the initial idea was to follow the method described in [45]. Being the package implemented for ROS and, therefore, not available for ROS2 projects it couldn't be employed in this work. To find a solution to this problem, an alternative way to assess the geometric characteristics of the terrain surrounding the vehicle had to be defined. The chosen approach is conceptually simpler than the initial one, but is still able to gather the information needed to make an efficient evaluation. From the range sensor used to obtain point cloud data, the approach is able to estimate, for every point, the slope of the terrain allowing the rover to understand the morphology of the soil. This empowers the vehicle with the ability of detecting both positive and negative obstacles, represented by abrupt changes in the slope.

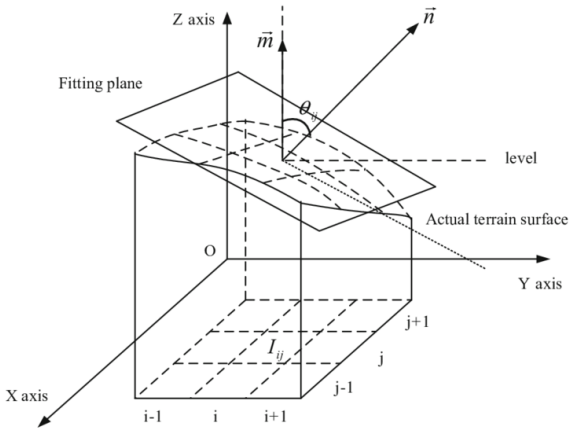

Figure 4.17: Example of a slope computation method[58].

Structure of the approach

The method, conceptually similar to the above figure, is articulated into four steps:

1. **Creation of the point cloud**, this process is completed using the library *Open3D*. The important preliminary steps to obtain the correct point cloud to process are two transformations, since the initial result obtained will be in the camera frame, but oriented according to the *Open3D* default settings. The first transformation is needed to rotate the axes of the frame and obtain the default orientation of the camera in Oaisys, which has the x axis to the right,

the y axis upwards and the z axis inwards. The second transformation will be used to go from the coordinates of the cloud in the camera frame to the coordinates in the base-link frame (coincident with the world frame, being the approach in use robot-centric), a term used to indicate the frame which has its origin in the center of the rover and its xyz axes respectively to the right, forward and upwards.

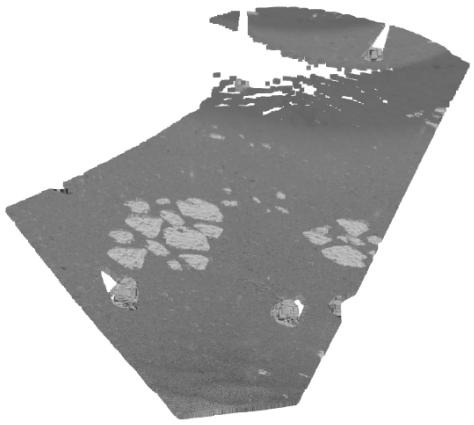

Figure 4.18: Sample of a transformed point cloud.

2. **Normal vectors estimation**, the transformed cloud object from the previous point can benefit from a method called *estimate_normals*. Differently from figure 4.17, this method, for each point in the cloud, uses a hybrid search algorithm, combining K nearest neighbour and radius search, to find neighbouring points to employ in the principal axis computation through covariance analysis. As reported in the *Open3D* documentation, the covariance analysis algorithm produces two opposite directions as normal candidates. Without knowing the global structure of the geometry, both can be correct. This is known as the normal orientation problem. *Open3D* tries to orient the normal to align with the original normal if it exists, otherwise, it does a random guess. To manually help the process, it is has been employed the *orient_normals_to_align_with_direction* method. Passing it the vertical axis $z = [0,0,1]$, as shown in figure **??** and **??**, allows to isolate only the outgoing normal vectors, needed in the computations of the approach. At the end of

the process, the point cloud is enriched with an additional field containing the corresponding estimated normal vectors for all its points.

3. **Slope computation**, once the normal vectors are available the angle between the latter and the vertical axis represents the slope of the terrain in that specific point of the cloud. It can be obtained taking advantage of the dot product formula, being z the vertical axis and n the estimated normal vector:

$$z \cdot n = \|z\|\|n\| \cos(\angle(z, n))$$

then, it is possible to retrieve the angle as:

$$\cos^{-1}(\angle(z, n)) = \frac{\|z\|\|n\|}{z \cdot n} \tag{4.4}$$

4. **Cost map creation**, as per the appearance-based method, also in the geometry-based one a traversability cost map needs to be outputted. The costs have been, again, assigned from 0 to 1 accordingly to the slope of the terrain. A slope with, either positive or negative, higher magnitude represents a geometric deformation of the terrain and a proportionally higher traversability cost. To determine the cost assignment, a function has been employed to make the cost start from 0, in correspondence of a completely horizontal surface, and achieve the maximum value of 1, untraversable, for terrain portions with a slope greater than a certain threshold (t) value. To obtain a result which highlights more higher slopes a second key value, called soft threshold (t_{soft}), has been determined to separate a first, more gentle, portion of the function from a second, steeper, section, as shown in eq.4.5. The two values have been set to 30°, for the soft threshold, and 70°, for the threshold. They can be tuned depending on the setting in which the rover is expected to navigate and the structural characteristics of the vehicle. The selected values are just a generic example employed in this particular case. The same evaluation is done considering the cases in which the normal is inclined in the opposite direction, resulting in angles between 360° and $360 - t$°, while for possible cases of angles between t° and $360 - t$°, obtainable only for points with faulty normal estimation, the cost is always kept at 1 as an additional safety measure.

$$cost_{geom}(\theta_p) = \begin{cases} \frac{2}{\pi} \cdot \theta_p & \text{if } \theta_p < t_{soft}[rad] \\ \frac{1}{t[rad]} \cdot \theta_p & \text{if } t_{soft}[rad] \leq \theta_p \leq t[rad] \\ 1 & \text{if } t[rad] < \theta_p < 2\pi - t[rad] \\ 1 - \frac{(\theta_p - (2\pi - t[rad]))}{t[rad]} & \text{if } 2\pi - t[rad] \leq \theta_p \leq 2\pi - t_{soft}[rad] \\ 1 - \frac{(\theta_p - \frac{3}{2}\pi)}{\frac{\pi}{2}} & \text{if } 2\pi - t_{soft}[rad] < \theta_p \leq 2\pi \end{cases} \tag{4.5}$$

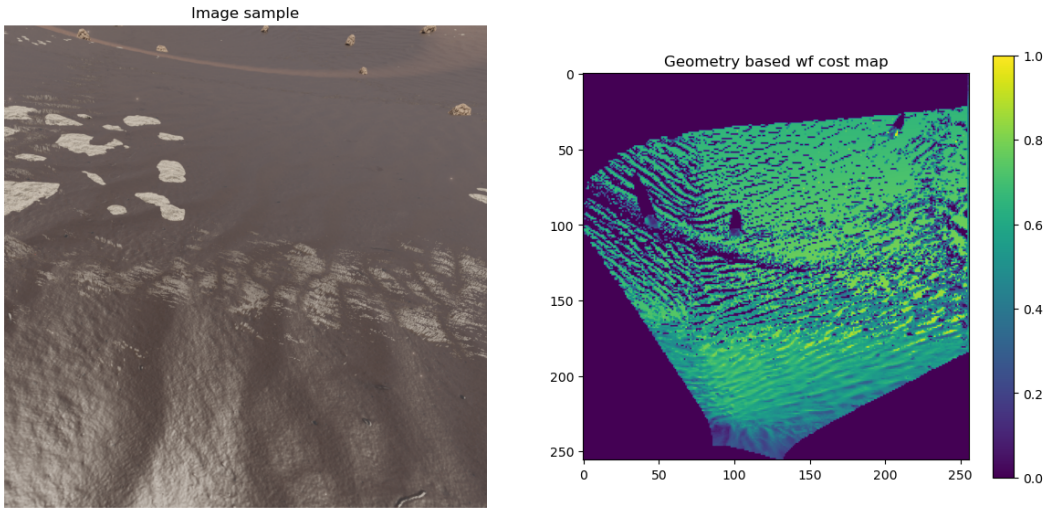

Figure 4.19: Sample of the geometry based cost map.

4.3 Evaluation metrics

In this section another really important topic concerning the training process of the neural networks employed will be covered, the evaluation metrics. They are fundamental in the understanding of the next chapter in which play a crucial role in all the experiments, performed during the development of the framework. They constitute a reliable measure in the assessment of the performances of the networks and represent the main indicator used to determine the satisfaction degree of a training procedure.
Having used, for the two the appearance-based modules, the same network, being both of them aimed at a classification task, the same metrics have been considered.

4.3.1 Loss function

The loss function, employed during the training procedure of a network, is the first, and one of the most effective, metrics to assess the quality of the training. Studying the curve of the function's values over the epochs, it is possible to see how much the learning capabilities of the network are effective. It represented by the magnitude of the loss decrease, in the graph, as the network trains. It is also possible to determine the correctness of the hyperparameters' setup and the constitution of the dataset, researching in the function's curve symptoms of either over or under fitting. These two phenomena, to be noticed, need the direct availability of a comparison between the loss curves of both the training and validation set, to see whether the validation loss is consistently above the training loss in the various phases, respectively the early stages for underfitting and the later stages for overfitting, of the training process.

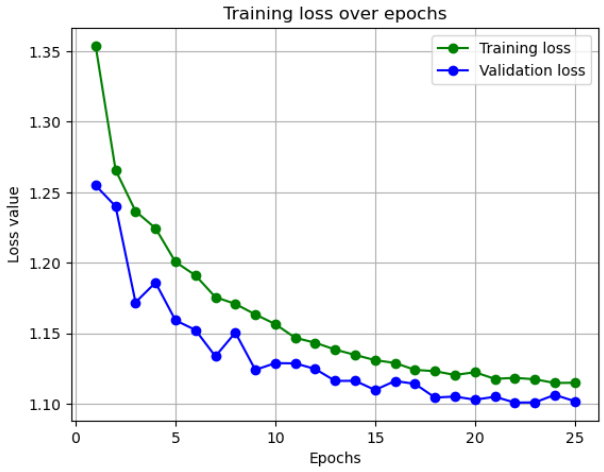

Figure 4.20: Example of correct losses' behaviour during training.

As mentioned in section 4.2.1, the loss function employed during the training procedure of both the two networks is a multi-class weighted cross entropy. It is a slightly modified version of a widely employed loss function for classification, with the adjustment of having the possibility to assign a weight to each class. As already covered, this is particularly useful in cases of unbalanced datasets, category in which the case of study falls perfectly, with the aim of amplifying the learning effect of the less represented classes.

4.3.2 Additional metrics

Other than the loss function, among the many possible choices, six metrics have been selected to be evaluated during the training procedure to prove the effectiveness of the learning process. All of them are implemented using the *torchmetrics* library, allowing to obtain metrics objects which can benefit from a series of useful methods. In particular the employed metrics are:

- **Accuracy**, measures, in multi-class classification, the percentage of correct predictions over the total number of predictions made for every class. In the equation 4.6 it is possible to see the formula used in accuracy computation, where, for N pixels of the ground truth belonging to class c, y_{ci} represents the exact label value, $\hat{y}_{ci}$ represents the predicted label and the result of the equality is either 1 in case of correct prediction or 0 otherwise. The metric is computed for both training and validation sets and is updated for every batch of every epoch, yielding, at the end of each iteration, its average value. Pixel accuracy on its own is, however, not enough to assess the performances of the network and, for this reason, other metrics with different characteristics have been employed too.

$$Acc_c = \frac{\sum_{i=1}^{N} 1 \cdot (y_{ci} = \hat{y}_{ci})}{N} \tag{4.6}$$

- **F1 score**, is the name chosen for the value of the harmonic mean of precision and recall, eq.4.7, enclosing the positive sides of both of them. Precision measures the quantity of true positive predictions (cases in which an element is correctly predicted in a specific class) over all the positive predictions (all the times an element has been predicted in this same class). Recall, on the other side, for every class, is the ratio between the true positive predictions over the total amount of positive samples (the total number of elements belonging to a certain class). The F1 score is able to balance both of them providing a single metric that reflects a model's ability to correctly identify positive instances, while minimizing false positives and false negatives.

$$F1_c = 2 \cdot \frac{precision_c \cdot recall_c}{precision_c + recall_c} \tag{4.7}$$

- **Jaccard index**, or Intersection over Union, is a metric, ranging from 0 to 1, considered as one of the most meaningful for image segmentation tasks [59]. It consists of the ratio between the intersection and the union of two sample sets (eq.4.8), in the specific case of semantic segmentation or image classification represented by the ground truths and predicted masks. It has been employed, as the previous metrics, class-wise to have as output a vector of values. Similarly to the F1 score, IoU too is capable of considering both false positives and false negatives making it more sensitive to segmentation quality than pixel accuracy, it is also able to consider the spatial distribution of the misclassifications, contrarily to accuracy. This means that, for example, if an object, like a rock in this specific setting, is slightly shifted from its true location, IoU can still capture the degree of overlap and provide a meaningful evaluation.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{4.8}$$

All the measures seen until this point have been relative to the different classes, but to have a more comprehensive evaluation it has been decided to consider also some overall metrics.

- **Dice score**, which is another name for the aforementioned F1 score, is computed in the same way, but in an overall manner. This is done with the aim of having a global view over both precision and recall.
- **Mean IoU**, as for the previous metric is linked with one of the earlier entries in this list, the Intersection over Union or Jaccard index. It represents the average value of the metric over all the classes in the classification task. As for its per class version, is one of the most important and widely used metrics for this kind of tasks and ranges from 0 to 1.
- **Confusion Matrix**, for multi-class classification is a NxN matrix, C, containing in its entries, $C_{i,j}$, the percentages of observations known to be in class i, but predicted to be in class j. As it is reported in the documentation of the library *torchmetrics*:
  - $C_{i,i}$, represents the number of **true positives** for class i.
  - $\sum_{j=1,j\neq i}^{N} C_{i,j}$, represents the number of **false negatives** for class i.
  - $\sum_{i=1,j\neq i}^{N} C_{i,j}$, represents the number of **false positives** for class i.

– The sum of the remaining cells in the matrix represents the number of true negatives for class i.

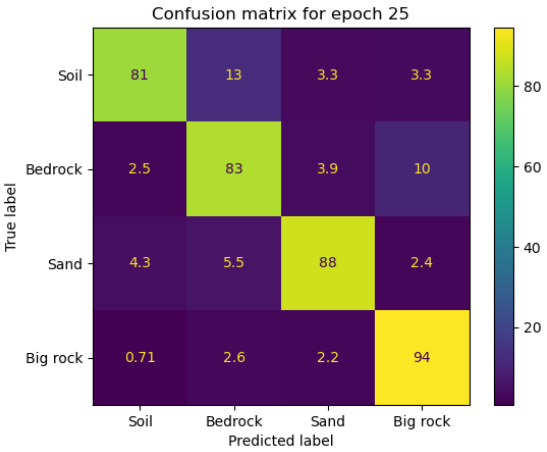

Figure 4.21: Confusion matrix sample of the segmentation task.

In the above paragraph, throughout the metrics' presentation, a series of terms has been used with respect to the predictions made by the network. When speaking about multi-class pixel level classification, it can be easier to understand them when viewed in the context of the classification ground truth and the predicted mask, as can be also seen in figure 4.22:

- **True Positive**, the area of intersection between Ground Truth(GT) and segmentation mask(S). Mathematically, this is logical AND operation of GT and S.
- **False Positive**, the predicted area outside the Ground Truth. This is the logical OR of GT and segmentation minus GT.
- **False Negative**, number of pixels in the Ground Truth area that the model failed to predict. This is the logical OR of GT and segmentation minus S.

Figure 4.22: Example of what is analyzed by segmentation metrics.

4.4 ROS architecture

The Robot Operating System is an open source software development kit for robotics applications [60]. In simpler terms, it is the ecosystem that enables all the pieces presented in the previous sections to work together and create the pipeline needed to carry out the desired results from the data acquired by the rover sensors. ROS is the basis for most robotic research from single-student projects to multi-institution collaborations or large-scale competitions and, considering its extremely active global community of millions of developers and users and its complete open source nature, embodies the qualities of a perfect tool to build the architecture formulated in this thesis work.

The fundamental principles of ROS are built upon few key concepts:

- **Packages**, as mentioned in [61],are the most basic units of the ROS software. They contain the ROS runtime process (nodes), libraries, configuration files, and so on, which are organized together as a single unit. Packages are the atomic build and release item in the ROS software.

- **Nodes**, are independent modules capable of interacting with each other making use of the ROS communication properties to exchange data. They constitute all the code which is executed while the system is running, if projected into the specific case of this work, for example, each approach will be implemented in a separate node.

- **Messages**, define the information sent and received during nodes' communications. Depending on the kind of data, specific type of messages exists, or can even be created ad-hoc if not already available for a specific task.

- **Topics**, constitute the channels over which data are continuously exchanged. They are like a tunnel in which a certain kind of message travels through. With respect to this channel a node can send something, being defined as a publisher on that topic, or receive something, being instead a subscriber to that topic.

- **Services**, differently from the communication established between nodes using topics, services are employed to define a kind of client-server interaction between processes. A service can be called from as many clients as desired and is used, usually, to either make computations or change some settings.

- **Launch files**, are a tool available in ROS and ROS2 which allows to startup and configure a number of executables containing ROS nodes simultaneously.

All of the presented building blocks, exception made for services, are employed in the ROS architecture defined for the presented framework, as can be seen from the figure below.

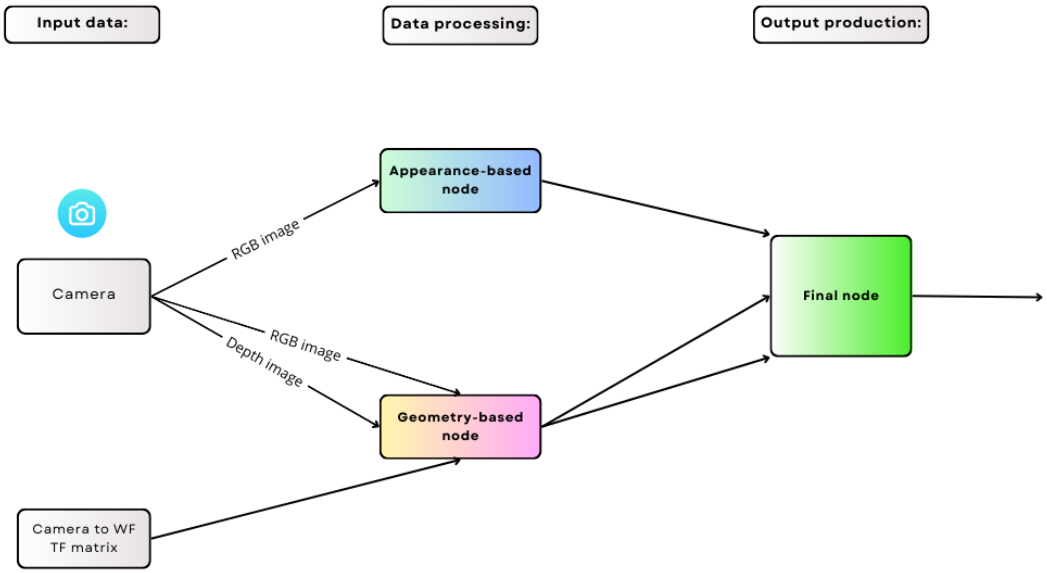

Figure 4.23: Complete ROS architecture.

The proposed ROS structure comprises 3 different nodes which will be started at once thanks to a launch file. They create the complete data flow which from the input data, coming from the rover, leads to the output of the final traversability cost map. In ROS and ROS2, which is the one specifically employed in this work, it is possible to create packages for nodes coded either in *C++* or *Python*. To keep a coherence with all the work presented up until this point and to be able to use the same libraries, all the nodes, which will be listed, and the launch file are completely implemented in *Python*.

4.4.1 Nodes' implementation

The information, coming from the camera on the UGV, will be sent to the two nodes in charge of the hybrid traversability analysis which will process them creating

the two, already discussed, traversability maps. The outputs of the nodes will go through an approximate time synchronizer, inside the final node, before being employed for the creation of the hybrid cost map. This last node will be in charge of the remaining processing operations on the maps and their combination, to get to the final desired result.

In the following paragraphs each part of the graph will be individually presented more in depth going through both the coding implementation, the logic behind the choices made and the interactions with the other elements of the pipeline.

Source data publishing

As can be seen in the far left of fig. 4.23, the data needed by the two approaches come from two sources, the camera and the rover itself. For what concerns the appearance-based side, the only RGB image is required and travels inside an *Image* message, from the *sensor_msgs* package. The geometry-based approach needs instead few more information, requiring also the corresponding depth image and the transformation matrix describing the pose of the camera with respect to the vehicle. The latter is not retrieved in the same way as for the images, but thanks to the use of a *Buffer* and *TransformListener* from the *tf2_ros* package, it can be obtained whenever its needed sending a lookup request, using the *lookup_transform* method of the *Buffer* object.

ApproximateTimeSynchronizer

Time synchronizers are a category of message filters, from the *message_filters* package, with the aim of synchronizing messages. The filter used in this node is an Approximate time synchronizer, which differs from the standard one since it considers a time range in which the message is accepted as synchronized. This kind of filters just checks the timestamps of the messages linked with it and, when they are acceptable, executes a callback, whose content is defined by the user.

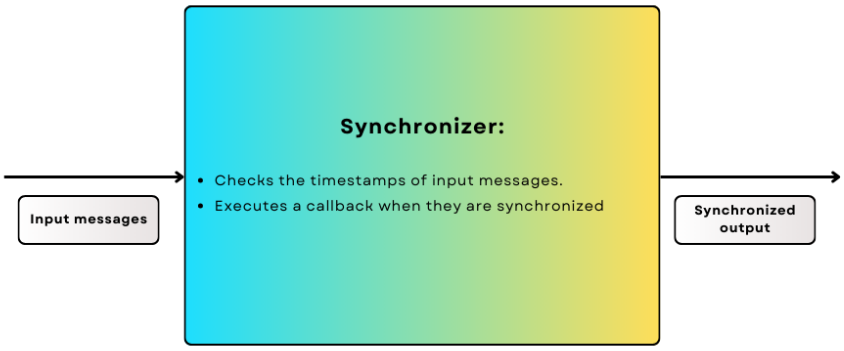

Figure 4.24: Data synchronizers.

Their aim is to synchronize all the inputs and then forward them to the successive steps. This phase is needed in order to be sure that the data processed together are actually meant to be together, since the two approaches could yield results not exactly at the same rate. They are employed inside two of the nodes in fig.4.23, the geometry based one and the final node. The first synchronizer is used to make sure that the three inputs of the node are consistent with each other, to avoid producing unexpected results. The second one, instead, ensures that the two cost maps of corresponding time instants, the relative pixel-grid correspondence, and the depth flags' matrix passed to the final node, are meant to be together, checking their timestamp. In figure 4.24, the steps operated in the filter node can be seen in detail, firstly the timestamps of the messages, received through *Subscriber* objects, are read and if considered synchronized, by how the *ApproximateTimeSynchronizer* is defined, the callback is executed. In this callback, all the intended processing operations are done being sure to work with the correct data.

Appearance-based node

This node, together with the geometry-based one, represents the implementation of the hybrid-approach discussed in the previous sections. Here, the two neural networks are employed to predict a semantic segmentation and a roughness classification masks, used to obtain the maps to combine into the appearance-based traversability cost map.

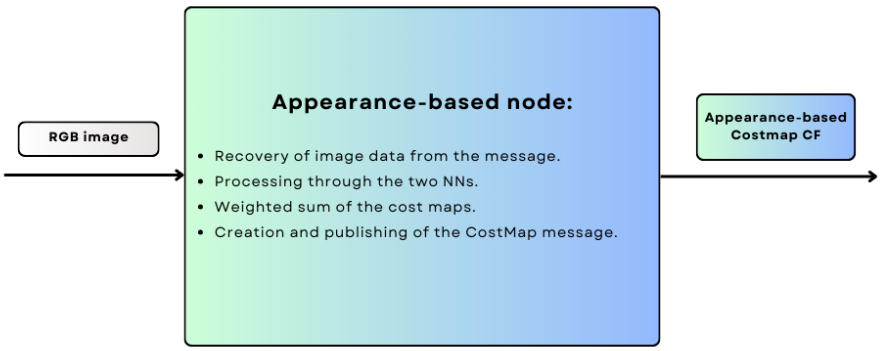

Figure 4.25: Appearance-based approach implementation.

The node class, defined for this case, has 8 attributes used to store all the data needed to complete the execution of the method. Among them can be found the device, either processor or GPU, to operate the network inference, the two networks' models for segmentation and classification, the weights to sum the two cost maps, the transformation object to apply to the received image to be handled by the networks, the subscriber to retrieve the images coming from previous steps and the publisher to propagate the resulting cost map to the next phase. Additionally, in the same ___*init*___ method as the attribute definition and initialization, the checkpoints containing the state of the networks after being trained are loaded, in order to work with trained instances and use them only for inference.
Since to employ the method the input RGB image needs to be available, the whole

approach is implemented in the subscriber's callback, activated when the latter, acquired by the camera, is passed to the node:

- ***app_callback***, as soon as the *Image* message, from the *sensor_msgs* package, is received, its content is extracted and restructured into a three-dimensional array with a bi-dimensional layer for each of the three color channels. This process is required due to the fact that the data travel as a flattened mono-dimensional array, while the dimensions of the image are contained in the height and width fields of the message. To this new object the transformation is applied, to resize it, normalize its values in the range 0-1 and turn it into a tensor object. Then, a fourth dimension is added to the previous result, since the networks are used to work with four dimensional batches of images' tensors. Finished all this pre processing, the inference of the networks is evaluated and the two cost maps are retrieved. The semantic segmentation traversability map and the roughness classification one are then combined using one of the most common strategies, a weighted sum of the two is computed. This allows to be able to manually determine which traversability estimation we want to value more, assigning a higher weight in the sum. In this particular application, as anticipated in previous sections, the roughness classification is employed as a refinement measure with respect to the prediction of the semantic segmentation network, thus resulting in a higher cost for the semantic cost map and, instead, a lower one for the roughness traversability evaluation. As presented in [49], and visible in the figure 4.26, a way to find the pixel corresponding to each cell in the grid map could follow a projection process based on the availability of an elevation map of the surroundings of the rover. Being the proposed framework substantially different from the one described in [49], for what concerns the geometry-based approach, a different strategy is employed. As will be described in the next points, it is a conceptually simpler method relying on a particular characteristic of the *Open3D* library, used in the geometry-based side, but obtaining an equivalent result.

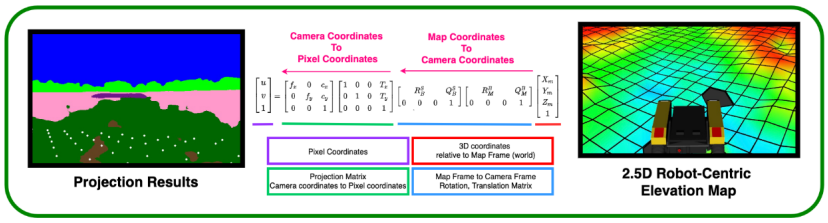

Figure 4.26: Projection process exposed in [49].

Therefore, satisfied with the obtained cost map, some final operations are executed before publishing the result to be used in the next steps. Since the

message type to be used is the *OccupancyGrid*, from the *nav_msgs* package, the costs need to be translated from the 0-1 range into the accepted format for these messages, a 0-100 range of integers where 0 means free and 100 completely occupied and so untraversable. Last step consists in the flattening process of the map, into a list of integers, and the creation and publishing of the message.

Geometry-based node

This second node works in parallel with the appearance-based one to yield the traversability cost map depending on the geometric evaluation of the landscape. The node, is in charge of generating the point cloud data and applying the method presented in section 4.2.3, operating a traversability analysis relative to the terrain slope.

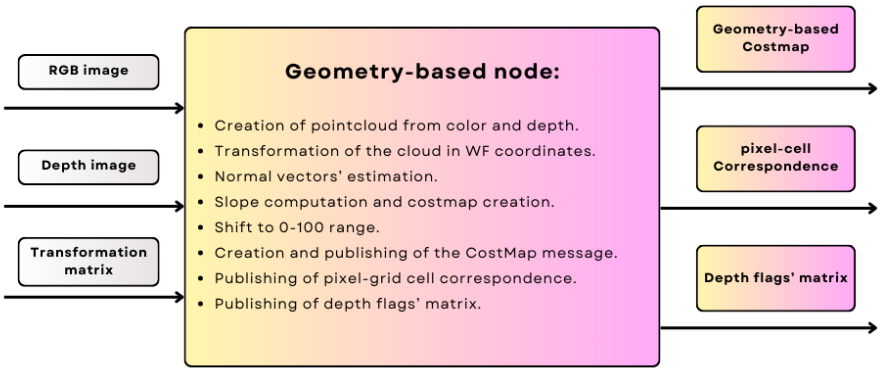

Figure 4.27: Geometry-based approach implementation.

Differently from the implementation previously described, this node has 18 attributes, due to the higher number of operations to be executed. Being the transformation data from the camera frame to the world frame broadcasted from the simulation, the node needs some way to be able to access those data whenever

it needs to. The solution to this task, as briefly mentioned in the data sources' paragraph, is represented by the definition of two of those attributes: a Buffer, from the *tf2_ros.buffer* library, to store the data during a certain specified time interval, and a TransformListener, from *tf2_ros.transform_listener* library, in charge of accessing the buffer and reading the transformations from it. To retrieve the correct transformation matrix other two attributes are defined, two strings representing the ID_s of the coordinate frames involved in the transformation. Other attributes must be reserved to the intrinsic parameters of the camera acquiring the images and the *PinholeCameraIntrinsic* object, used in the creation of the point cloud. Among the remaining ones can be found a preliminary transformation matrix to obtain the default camera orientation, as anticipated in 4.2.3, the subscribers to receive the RGB and depth images, an *ApproximateTimeSynchronizer* to be sure to have the correct images to process, and the three publishers which will propagate the geometry-based traversability cost map, the pixel-grid correspondence and the depth flags' matrix to the next step in the pipeline. For what concerns this last matrix, it has the same dimensions of the RGB and depth images, employed in the process, but its entries are either 0 or 1 depending on the satisfaction of a depth check. It's purpose is, in fact, to signal to the final node which pixels in the acquired sample are below a certain depth threshold, which must be defined as another attribute of this node.
As was for the appearance-based one, the totality of the method is enclosed in a callback, specifically the synchronizer's callback, activated as soon as the corresponding RGB and depth images get to the node:

- ***geom_callback***, here the geometry-based method is implemented. From the two messages, belonging to the type *Image* from the *sensor_msgs* package, the data of the RGB and depth images are retrieved and restructured into numpy arrays with the right dimensions, using the node's method *restructure_image*. Additional processing operations need to be applied on the depth image, to have the values in m_s, being forced to travel in mm_s in the *Image* message. From these depth data the matrix of depth flags is built, starting from a matrix of zeros and changing all the pixels under the depth threshold to 1. The value of the latter has been set to $10m_s$ with the aim of building a cost map as informative as possible. As can be seen in fig.4.19, the data get sparser and sparser as the depth increases, with a detrimental effect also on the employed traversability estimation methods, therefore being less accurate and informative for the completion of the task. The two images are converted into *Image* objects of the *Open3D* library and employed to generate a RGBD image. In the creation of the latter it can be specified the depth threshold to consider, with the *depth_trunc* parameter, to easily account for it in the geometry based method. The consequence of this choice will be a point cloud with less points, resulting also in lighter computations. The RGBD image,

together with the camera object, attribute of the node, is given as input to the *create_from_rgbd_image* method which will, finally, yield the point cloud object.
Once the preprocessing on the data is done, the approach, as it was previously presented, is applied. A lookup request for the transformation matrix is sent. In case the desired transformation matrix is not available yet in the buffer, the request process is inserted into a retry mechanism, built within a *while* loop. It will try again a certain amount of times, in case the correct transformation is broadcasted in the meantime, if it never arrives an exception is raised. Once a positive result is obtained, the first operation to be done is the definition of the 4x4 transformation matrix itself. This kind of matrices are constituted as follows:

$$T = \begin{bmatrix} r_{0,0} & r_{0,1} & r_{0,2} & t_0 \\ r_{1,0} & r_{1,1} & r_{1,2} & t_1 \\ r_{2,0} & r_{2,1} & r_{2,2} & t_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.9}$$

where $r_{i,j}$ entries define the rotation component of the transformation, while $t_i s$ describe the translation.
This data need to be restructured from the content of the broadcasted messages of the type *TransformStamped*, from the *geometry_msgs* package, which carry the information separating the translation data, kept as they are, and the rotation data, in the shape of the equivalent quaternion to the rotation matrix. To this aim, a *restructure_tf* function has been designed. For what concerns the t vector, no operation is needed other than extracting it from the message. For the R matrix instead, the following formula had to be implemented to retrieve the matrix entries from the quaternion elements:
being $q = (q_0, q_1, q_2, q_3) = (q_w, q_x, q_y, q_z)$, in terms of the content of the message, then

$$R(q) = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2\,(q_1\,q_2 - q_3\,q_0) & 2\,(q_1\,q_3 + q_2\,q_0) \\ 2\,(q_1\,q_2 + q_3\,q_0) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2\,(q_2\,q_3 - q_1\,q_0) \\ 2\,(q_1\,q_3 - q_2\,q_0) & 2\,(q_2\,q_3 + q_1\,q_0) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{4.10}$$

The transformation matrix representation is essential to the reference frame's transformation needed to be applied to have the geometry-based traversability cost map in the coordinates of the target frame. This frame, as mentioned previously is the robot's base-link, but will be equivalently referred to also as the world frame. Obtained the matrix, the *transform* method of the point cloud object is applied two times to go firstly in the default camera frame and successively in the world frame.
The next steps in the geometry-based approach are the estimation of the normal vector of each point, the manual selection of the only outgoing vectors,

as anticipated in previous sections, and the computation of the corresponding slopes, angles between the normal vector and vertical axis. From the slopes the geometry-based traversability grid map is created and, together with it, a correspondence between the pixels of the rgb image and the cells' coordinates is derived. For the same reason explained in the above section, the cost map is casted into an equivalent version comprehending integers in the range 0-100. These data are flattened out and inserted into an *OccupancyGrid* message, in which other information must, in this case, be added. It must be specified, the id of the frame in which the map is represented, the cells' resolution in meters, the width and height of the grid in terms of cells and the complete pose of the origin of the map itself. For what concerns the correspondence data and the depth flags' matrix, they become the content of two Image messages with appropriate dimensions and encoding. Finally, all three messages are published towards the next phases of the process.

Final node

The last node in the pipeline is the one in charge of gathering all the results and elaborate the final traversability cost map combining them.

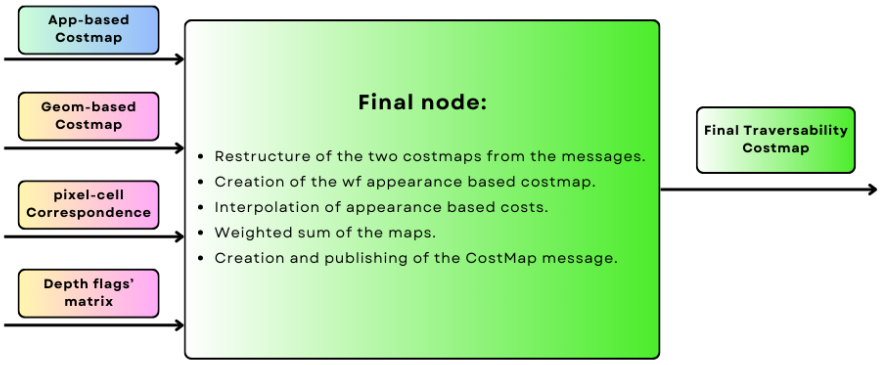

Figure 4.28: Final node of the architecture.

For what concerns the implementation of the ROS2 node, 10 attributes are defined. They are designed to store, among them, the four subscriber objects employed to receive the data needing synchronization (the two cost maps, the pixel to grid correspondence and the depth flags' matrix) and the *ApproximateTimeSynchronizer* itself. Two other attributes need to be defined to accomplish the transformation process of the appearance-based map, still in the image plane, a transformation object which will be used to resize to the correct dimensions of the cost map and the interpolation radius, required in the last phase of the operations leading to the wf appearance-based cost map. The last three attributes of the node are reserved to store, the two weights for the combination of the cost maps and the publisher in charge of handling the final hybrid result.
As for the other nodes, all the procedures are articulated in the implemented callbacks, in this case, similarly to the previous geometry-based node, the synchronizer's callback:

- ***final_cmap_callback***, being a filter's callback if and only if all the necessary data are arrived and synchronized the complete process is executed. The first thing done is to extract the content of all the input messages. For what concerns the appearance-based cost map, the pixel-grid correspondence and the depth flags' matrix, the data field of the messages are retrieved and restructured into a matrix. From the geometry-based cost map message, in addition to the aforementioned process, the parameters characterizing the grid, frame id, cells' resolution, position and orientation of the origin are gathered too from its content.
  To start back from where was left in the appearance-based node, the next step of the node's execution is, from the 2D image plane cost map, to obtain the relative grid map in the world frame. As anticipated previously, this is accomplished taking advantage of a specific characteristic of the *Open3D* library, the fact that the points' array, yielded by the point cloud object, is a mono-dimensional array ordered from the point corresponding to the top left pixel of the RGB image to the pixel at the bottom right. Knowing this correlation it is possible to restructure the array obtaining a correspondence between pixels and points' coordinates, which allows to derive the relation between pixels and cells of the grid map to which each point belongs in the geometry-based map. Being both the geometry and appearance based approaches applied on the same RGB image, this correspondence can be employed to create the grid version of the appearance-based cost map, without the need of elaborating a projection process, like the one in fig.4.26. Since the geometry based approach works with full resolution images, while the appearance-based side resizes them to be processed by the networks, the correspondence must be resized too to match the dimensions of the 2D image

plane cost map. The only addition to this, comes from the depth threshold imposed in the geometry-based node, to be sure to build the appearance grid map only for pixels in the allowed range the depth flags' matrix is resized too in the same way. After the resizing, a for loop is employed to select, for each pixel with corresponding positive depth flag, the relative cost in the map and assign it to the matching grid cell. Being the image taken by a camera, the perspective of the device causes the majority of the pixels to be dedicated to the closest portion of the grid map. The assignment of multiple points to the same cells, is taken into account by the computation of the average cost for the cells populated by more than one point. Another consequence, of the aforementioned condition, is the anticipated increasing sparsity in the map as the points get further from the camera. This phenomenon could be already seen in the geometry-based map in fig.4.19, but is even more evident in the appearance-based one, as can be noticed in the figure below.

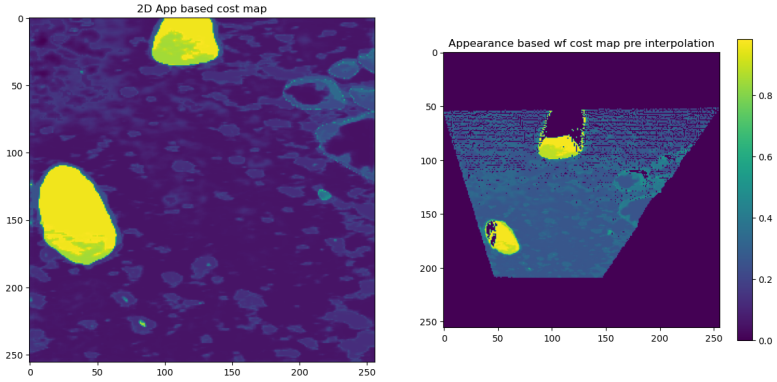

Figure 4.29: Appearance-based grid map sample.

To overcome this condition, which will cause incorrect behaviours in the combination of the cost maps, an interpolation process has been developed. For all the points at zero in this cost map and not in the geometry based one, an interpolation window with tunable dimensions is considered. All the pixels in the window are checked and their value is summed to the central one. The average, computed with respect only to the nonzero pixels, is evaluated and will become the new cost of the initial pixel. The process is repeated until the number of candidates is down to zero. In the case of an unsuccessful iteration, the size of the window is increased by five pixel to include more points and have a higher probability of finding a nonzero value. This can be re-iterated, if necessary, for a maximum of five times to avoid making the process too heavy. Even if the method is very simple, the result, which can be seen in the next figure, appears to be more than acceptable and coherent with what expected.

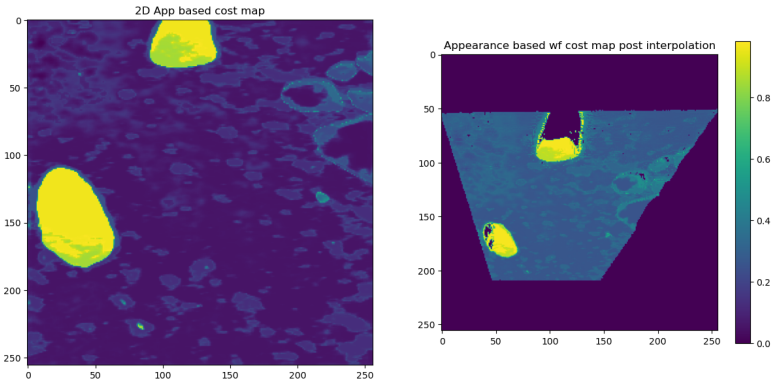

Figure 4.30: Appearance-based grid map sample post interpolation.

Now that all the data needed to elaborate the final hybrid traversability cost map are available, a combination of the two maps is computed through a weighted sum. The newly obtained result represents the fusion of the contributes coming from the two approaches, which try to compensate the possible flaws of each other. This map is, then, converted in the 0-100 range, flattened again and published inside a last *OccupancyGrid* message. All the characterizing information, retrieved from the geometry based map, are specified again for the final one, to be used by the other actors involved in the autonomous navigation system.

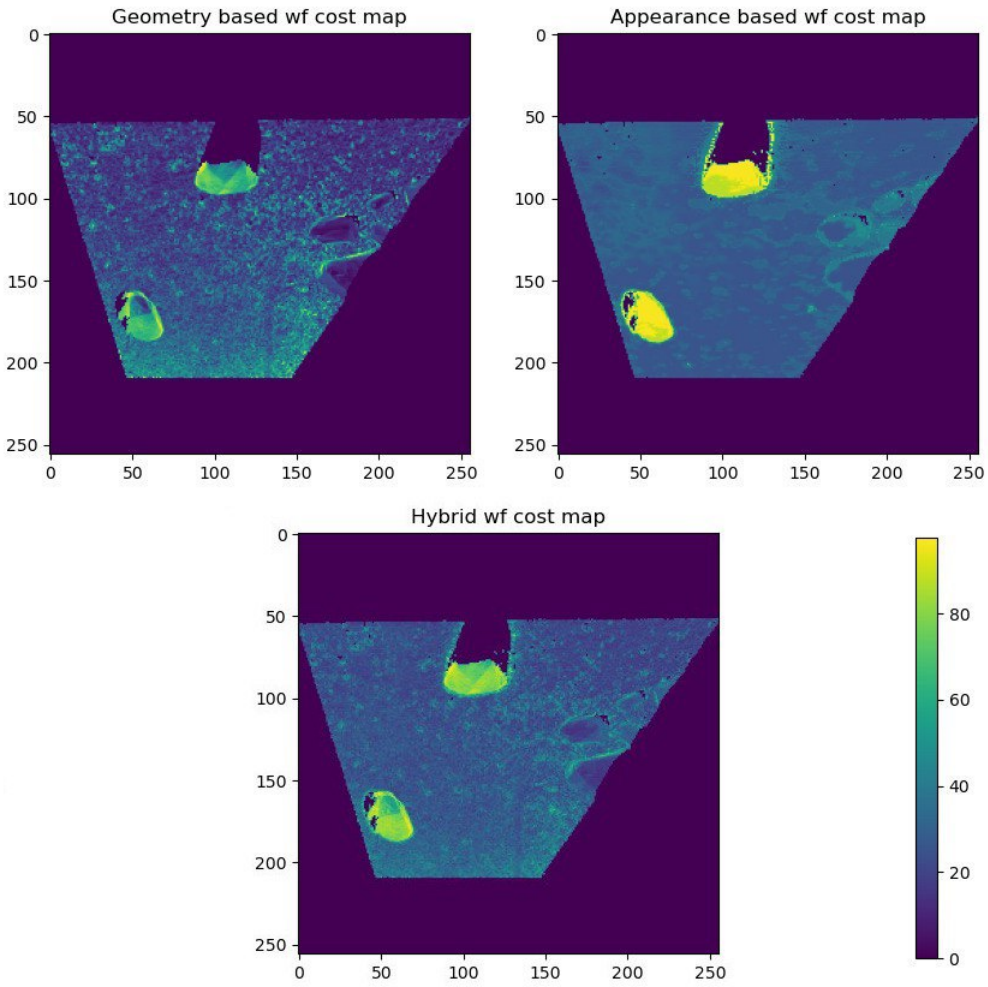

Figure 4.31: Hybrid terrain traversability cost map sample.

Chapter 5

Developments and Results

In this chapter, an overview of the main development phases is presented, highlighting, at first, the most important steps and the motivations which led to the final setup and, in the end, the actual simulation's tests to verify the performances of the framework.

5.1 Preliminary tests

The majority of the effort in the creation process was directed towards the appearance-based approach. After having defined the network to employ, the first step had been to test its effectiveness on the available AI4Mars dataset, limiting to the semantic segmentation side due to the unavailability of the other necessary information.

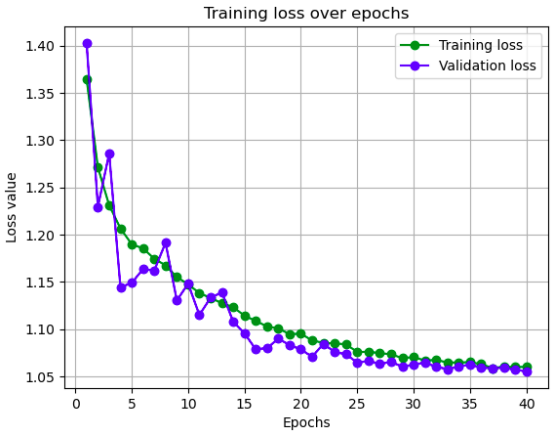

Figure 5.1: Plot of the losses during a training with AI4Mars.

As can be seen in the figure above, exception made for some peaks in the validation

loss curve, which can normally verify sometimes, the observed losses behaviour is symptom of a successful training procedure and such conclusion is confirmed by the values exhibited in the confusion matrix:

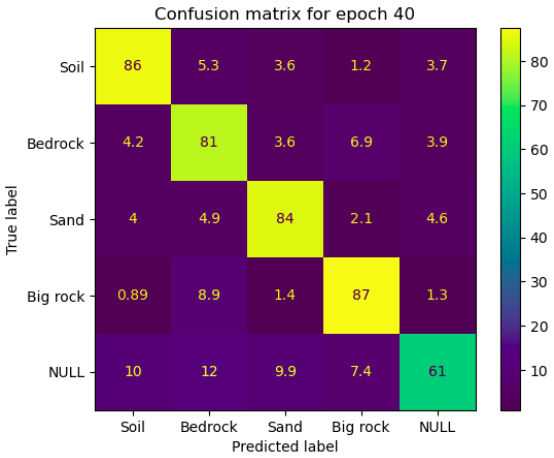

Figure 5.2: Last confusion matrix of the AI4Mars training.

As an effect of the weighted cross entropy, it is possible to notice two details: the first one is that, at least from the confusion matrix point of view, the unbalance of the **big rock** class had been compensated and the second one is that, on the contrary, the most represented class is the one with the lowest performances.

However, the assumption just made couldn't completely be verified observing the predictions of the network on test set's samples. As anticipated in section 4.1.1, due to the nature of the dataset itself, the test set presents a lot of images with just minor portions of the terrain labeled and even some completely unlabeled samples. The network, instead, tried to predict as much as it could resulting in cases like the ones which can be seen below:

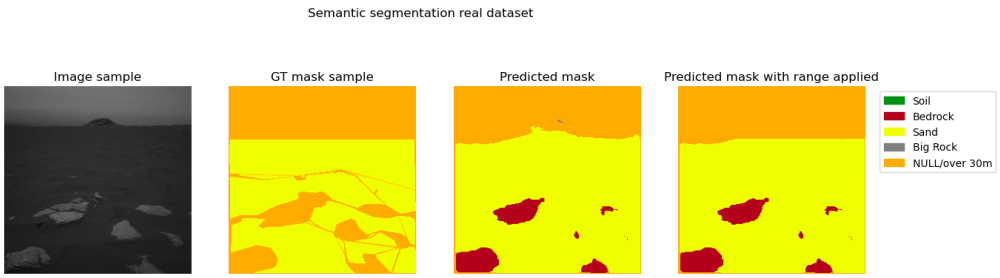

Figure 5.3: Test set sample with missing labeled terrain portions.

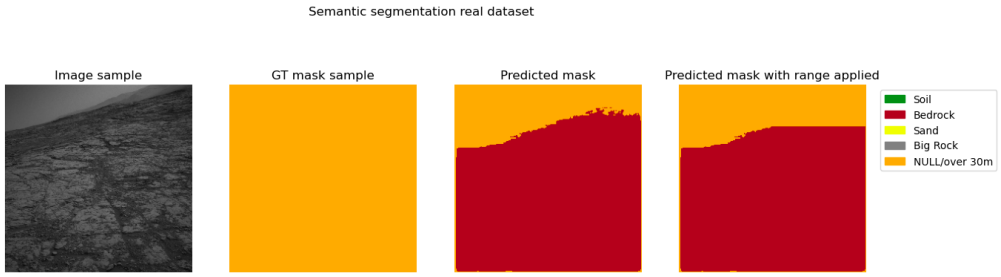

Figure 5.4: Test set sample completely unlabelled.

Setting aside the issues of the dataset, the result of the training was a signal of the correct functioning of the network, which allowed to proceed to the next phase, the switch to the synthetic set.

Its first version was a small dataset of about 200-300 images, to be divided in the three sets. Being in the initial steps of its creation, a smaller size was preferred due to the high time request to rebuild from scratch bigger collections, situation which frequently occurred due to the repeated modifications needed by the configuration file. As could be easily imagined, the smaller dimension had a visible effect also on the performances of the training process:

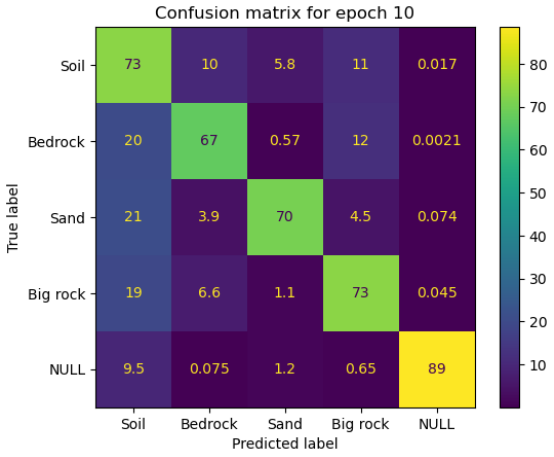

Figure 5.5: Last confusion matrix of the first synthetic training.

even if the values in the matrix were not bad, from a dataset with perfect labeling it is normal to expect really high fidelity in the networks' predictions. This appears even clearer looking at a sample of the predictions made on the test set, it is possible to see in the figure below that the result is far from being acceptable. The

smaller size of the dataset was, evidently, affecting the generalization capabilities of the network and the initial textures, employed for the different materials, were only worsening the quality of the predictions.

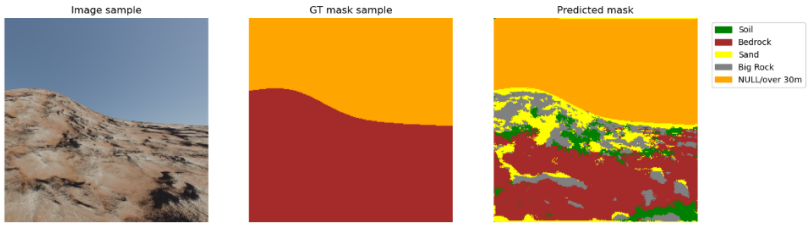

Figure 5.6: Sample of network's prediction on the test set.

The next important step was, therefore, the creation of a second version of the dataset with a size suitable for the training of a segmentation neural network. From approximately 300 images it was doubled to about 600, increasing significantly the performances at the end of the training.

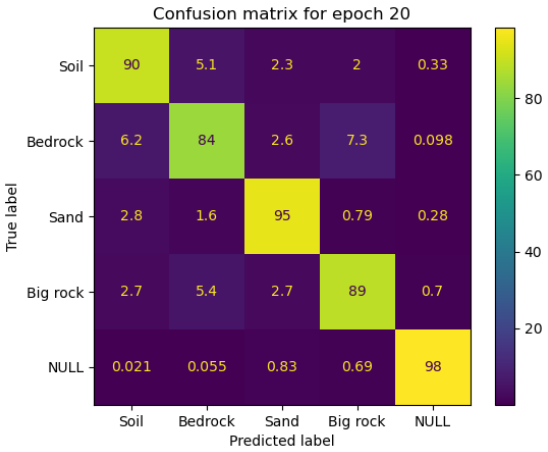

Figure 5.7: Last confusion matrix of the second synthetic training.

Comparing this result with the previous one, is clear how this second version of the set increased the aforementioned generalization capabilities. The network was able to predict with really good outcome all the terrain classes, overcoming the problem of having the most represented class showing worse results encountered

with AI4Mars. Looking at a prediction sample on the test set, is even more visible how, just modifying one parameter, the quality of the predictions changed dramatically obtaining almost a pixel perfect reproduction of the ground truth.

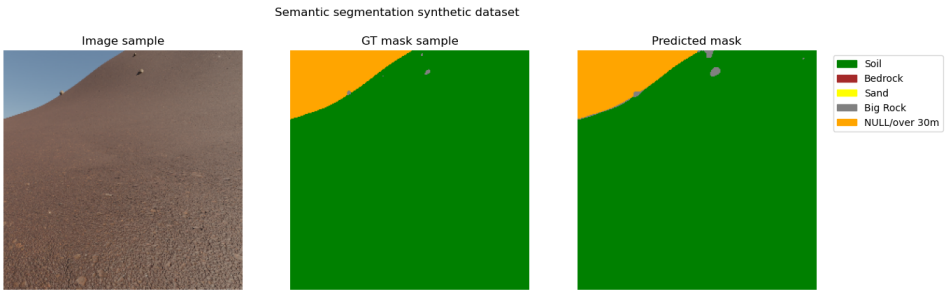

Figure 5.8: Sample of network's prediction on the test set.

In both of these two earlier versions, all the original five segmentation classes are present, but like already mentioned previously, a further change in the dataset resulted necessary when roughness estimation had to be involved. The presence of the sky, corresponding to the **NULL/over 30m** class, resulted in a lot of problems in the creation of the roughness ground truths. The easiest way to solve it was the generation of a third version of the synthetic set in which the orientation of the camera had been controlled to have the sky always out of the frame. Also the sand texture of this new version had been slightly changed, since the dataset was continuously updated trying to find the most resembling appearance possible to the real Martian soil.

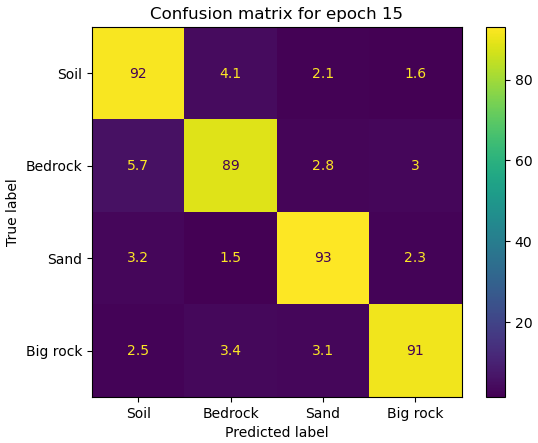

Figure 5.9: Last confusion matrix of the third synthetic training.

As could be imagined the performances of the network remained at least at the same degree, even improving the previous version in some cases, most likely due to the segmentation task becoming simpler having lesser classes among which differentiate.

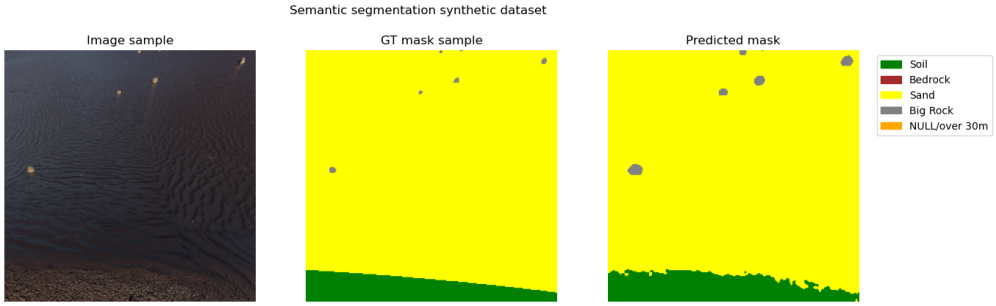

Figure 5.10: Sample of network's prediction on the test set with new sand texture.

As said earlier, this third version's purpose was the creation of roughness ground truths to be used in the roughness classification process. Even if the semantic segmentation performances were remarkable the opposite verified for roughness classification.

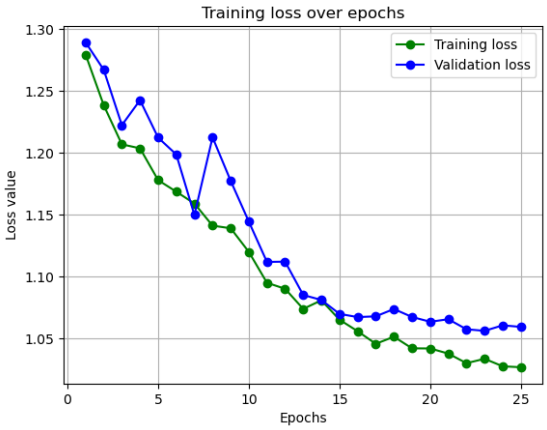

Figure 5.11: Plot of the losses during a first training of the roughness classification network.

In the above figure, it is possible to notice that, not only the decrease in the losses value is lower, even with respect to the AI4Mars training, but the validation loss was almost consistently above the training loss, showing clear signs of overfitting. This poor behaviour reflected, obviously, also in the quality of the predictions made on the test set:

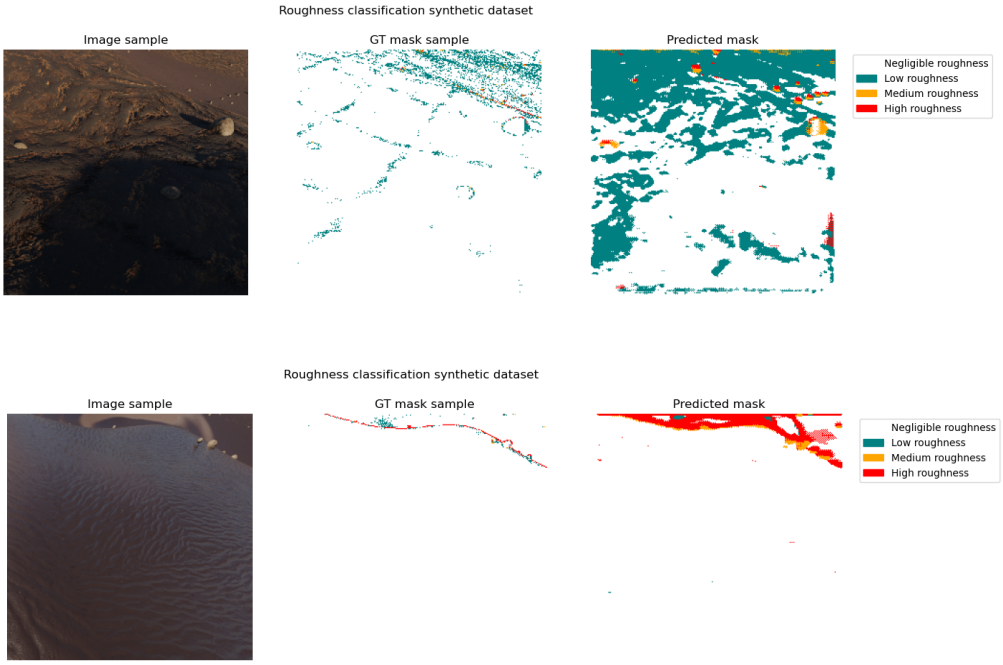

Figure 5.12: Samples of networks' roughness predictions on the test set.

The main reasons behind the low quality of the training are, again, to be found in the dataset employed. Its first iteration was extremely unbalanced, far more than what verified for semantic segmentation, being almost entirely constituted by pixels belonging to the **Negligible roughness class**, as can be also seen in the ground truths and predictions above. Another problem which required to be handled can be noticed in the first picture, where strange rectangular artifacts appeared in the textures and altered the results.

To fix as much as possible both of the previously listed flaws, a fourth and final version of the synthetic dataset was created. The modifications introduced were significant and affected the overall appearance of the simulated landscape itself. Starting from the more general and moving towards the more punctual changes, the preset employed to generate the environment was modified, as mentioned in section 4.1.3, trying to obtain smoother hills and depressions, realizing a morphology more similar to what observed in the real images of Mars' surface. The next step was the complete deletion of one of the textures, specifically the **bed rock** one, substituted with custom blender objects more representative of the scattered stone slabs constituting that class. The advantage obtained from this choice was not only aesthetic, but also functional from the roughness point of view. Those objects, in fact, can be modified more easily than a whole texture, and present a more rugged

and geometrically complex surface than the previous employed material. The **big rock** class was also reworked, switching the smoother rocks used initially for more realistic ones created, again, ad hoc. As can be seen in the next figure, these changes resulted in a still unbalanced, but more uniform situation, significantly differing from the starting point.

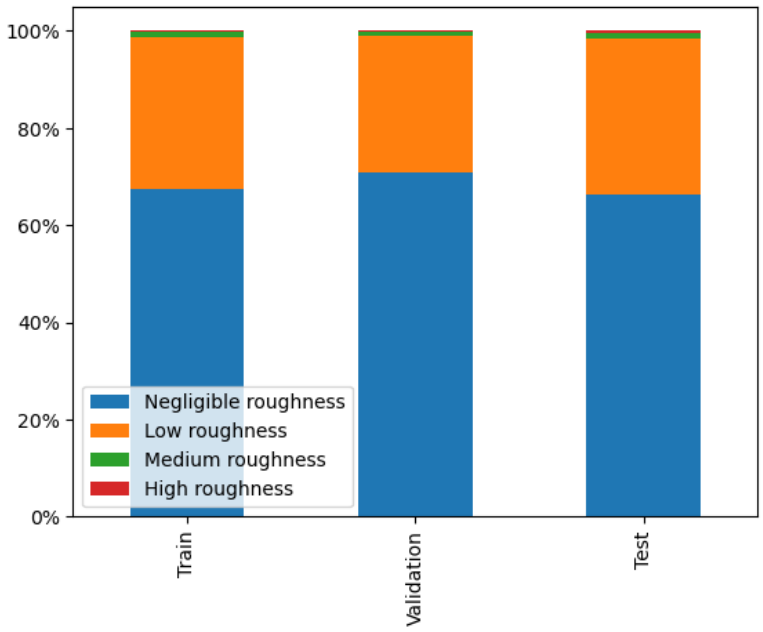

Figure 5.13: Frequencies of the roughness classification dataset.

The last changes introduced were more subtle and oriented to both fixing the problem of the rectangular artifacts and trying to optimize as possible the images acquired. To achieve the first goal, it was needed to modify the *size* and *dispStrength* parameters of the two remaining textures since the artifacts appeared only at the borders of their repeated patches. Those parameters control respectively the dimension of the texture tile positioned on the terrain portion dedicated to that material and how much the displacement map affects the geometry of the texture. After some empirical tests, a trade-off has been found with the consequence of the artifacts seeming to be vanished, or at least appearing really rarely.
This fourth version of the set confirmed the quality of the semantic segmentation network, even improving what obtained before, as can be noticed in the following figures.

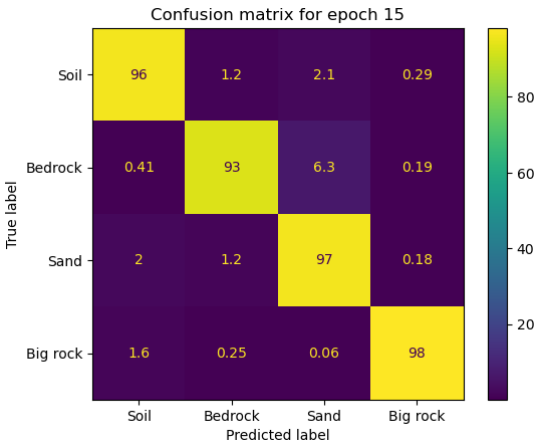

Figure 5.14: Last confusion matrix of the fourth synthetic training.

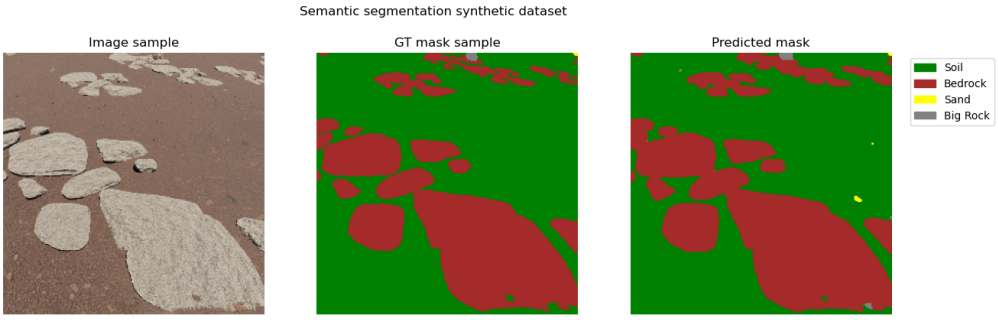

Figure 5.15: Sample of network's prediction on the test set with new materials.

For what concerns the roughness classification, it must be highlighted how the overfitting problem has been solved:

Figure 5.16: Plot of the losses during the training of the accepted roughness classification network.

the validation loss stays , almost entirely, below the training loss, behaviour which was never achieved in the previous case. Looking at the confusion matrix, in the figure below, the achieved performances appear not as good as was for the segmentation, but it needs to be taken into consideration the significantly higher level of the task's difficulty. It has been asked, to this network, to predict a physical property of the soil just by looking at a picture, which ,even if instinctively simple for human beings, is much more harder, from a deep learning point of view, than differentiating between the various terrain classes.

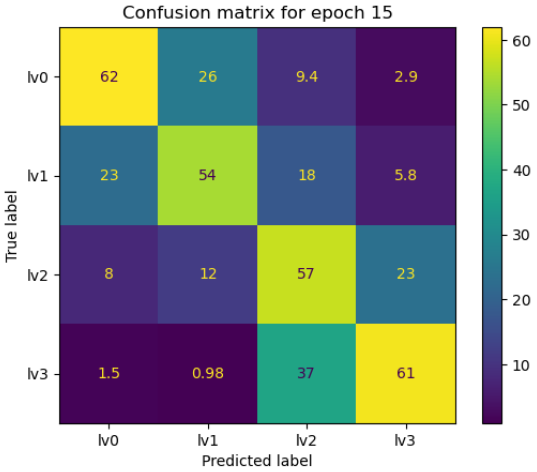

Figure 5.17: Last confusion matrix of the new roughness classification training.

For this reason, the most important result to take into account is the quality of

the predictions, to be sure that the results can be considered acceptable.

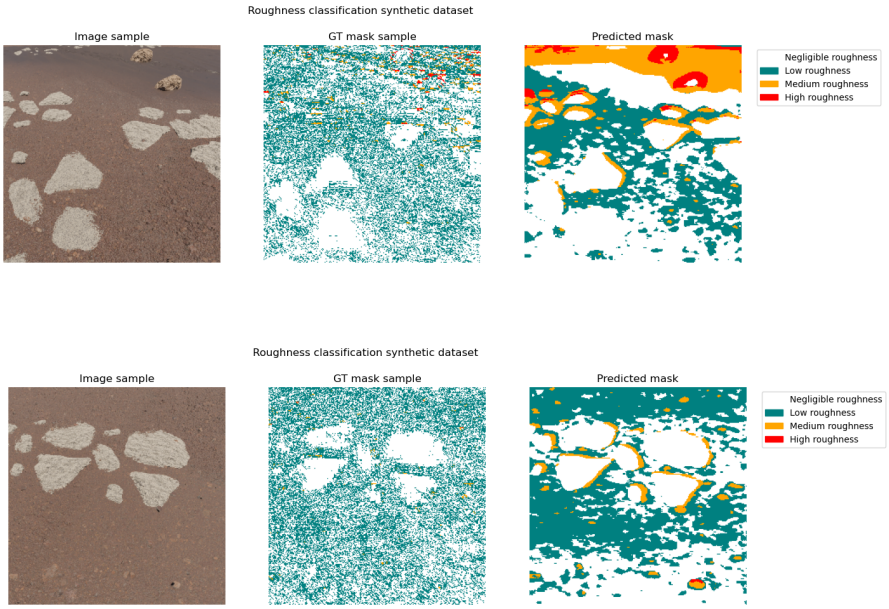

Figure 5.18: Sample of network's roughness prediction on the test set with new materials.

As can be seen in the above figures, the performances in the predictions can easily be considered acceptable. Even if the network is far from achieving a pixel level accurate reproduction of the ground truth, its main goal is accomplished. The different areas of the images have the correct depth levels, the only flaw which can be highlighted is a general over estimation of the roughness in some areas, mainly between medium and high roughness. Although this is, by all means, a defect, two points must be considered which help in understanding why the obtained quality has been accepted as satisfactory:

1. Roughness estimation is just a refinement measure for the semantic segmentation prediction. This means that the contribution of the roughness cost map to the final appearance based result is lower and used mainly to increase the detail level among areas belonging to the same terrain class.

2. The flaw which has been just pointed out, is only increasing the costs assigned to certain areas. This is, of course, not to be considered correct but, given the already mentioned high degree of task's complexity, can be accepted since will only cause to have an even more cautious planning. In particular, if it is considered that happens mainly between the highest and most dangerous roughness levels.

5.2 Final tests

5.2.1 Simulation setup

The idea at the core of this testing phase consists in simulating an actual working situation for the framework, in order to be able to evaluate its current performance level and online applicability. The setup of these test has been realized through the use of oaisys, in fact, as mentioned in [51], two approaches can be used to define the pose of the sensor acquiring the images inside the simulation: randomly selecting the characterizing values of position and orientation from given intervals, as done in the creation of the dataset to stimulate a higher variability in the samples, or by providing through the configuration a *.csv* file containing all the poses to be selected in a batch. In the aforementioned paper, this methodology is only cited without detailing the definition of the configuration file, which is provided, in the available material, only to be employed with the other approach. In order to adjust the configuration file to have the sensor following the desired behaviour, the section to be modified is the one called ***SENSOR_SETUP***, in particular the ***GENERAL*** subsection. As can be seen in figure 5.19, this section of the previously employed cfg file is reported. The main detail to be noticed is in the *movementType* and *SensorMovementType* fields defining in which way the simulator will handle the setup of the sensor. The argument *randomEuclideanTarget*, signals to oaisys that the already discussed method of setting up position and orientation of the camera at random from given intervals and forcing it to frame a target object has been chosen.

```
"SENSOR_SETUP": {
    "GENERAL": {

                "baseMovement": {   "movementType": "randomEuclideanTarget",
                                    "hoverBaseModeEnabled": true,
                                    "hoverBaseStage":"terrain01",
                                    "hoverBaseDistance":2.0,
                                    "hoverBaseDistanceNoise": 0.05
                                },
                "sensorMovementType":"randomEuclideanTarget",
                "hoverBaseModeEnabled": true,
                "hoverBaseStage":"terrain01",
                "hoverBaseDistance":2.0,
                "hoverBaseDistanceNoise": 0.05,
                "hoverTargetModeEnabled": true,
                "hoverTargetStage":"terrain01",
                "hoverTargetDistance":0.0,
                "positionOffsetEnabled":true,
                "randomEuclideanPosMin": [-50.0,-50.0,0.0],
                "randomEuclideanPosMax": [50.0,50.0,0.0],
                "randomEuclideanEulerMin": [0,0,0],
                "randomEuclideanEulerMax": [0,0,360],
                "randomTargetPosMin": [0,0,-4.84],
                "randomTargetPosMax": [0,0,-4.84],
                "randomTargetEulerMin": [0,0,0],
                "randomTargetEulerMax": [0,0,0],
                "randomTargetRollDeg": [0.0,0.0],
                "targetObjectActive":true,
                "tragetObjectMovementType":"random"},
```


Figure 5.19: Interested section in the previous oaisys cfg files.

In the figure 5.20 instead, the modified portion of the cfg file can be seen. All the lines related to position and orientation of both the sensor and the target have been deleted and there is a new argument, *deterministic*, for those two fields, to highlight the difference of the applied approach. The other relevant change is the *SensorMovementPose* field of the file, used to specify to the simulator where to find the poses' information.


```
"SENSOR_SETUP": {
    "GENERAL": {

                "baseMovement": {   "movementType": "deterministic",
                                    "hoverBaseModeEnabled": true,
                                    "hoverBaseStage":"terrain01",
                                    "hoverBaseDistance":2.0,
                                    "hoverBaseDistanceNoise": 0.0
                                },
                "sensorMovementType":"deterministic",
                "sensorMovementPose":"/home/user/Thesis/Code/test_csv1.csv",
                "interpolationMode": "none",
                "hoverBaseModeEnabled": true,
                "hoverBaseStage":"terrain01",
                "hoverBaseDistance":2.0,
                "hoverBaseDistanceNoise": 0.0,
                "positionOffsetEnabled":true,
                "randomTargetRollDeg": [0.0,0.0]},
```


Figure 5.20: Interested section in the modified oaisys cfg files.

Inside the *.csv* file, the poses must be reported on different rows and present eight entries each. The way to arrange the entries has been obtained, again, from the implementations inside the code of the simulator and respectively must be: an integer working as a kind of id, three floats to define the position of the sensor on the x, y and z axis and other four floats to define the orientation through a characterizing quaternion.
Some other slight adjustments have been made to the oaisys code itself, in order to have the hovering constraint still employed also in this other approach, keeping the sensor always at the same distance from the terrain, simulating the motion of a vehicle.

5.2.2 Tests' description

What has been described in the previous section is the baseline from where the simulation tests started. A *.csv* file, containing a series of consecutive poses, has been created to replicate the rover's motion in the environment. A starting location is randomly selected and the y coordinate is increased by a certain amount to represent the motion of the vehicle. A picture is acquired each time the navigation is completed gathering the new data to employ for the next step of the UGV. What has been done, in this case, is the generation of all the data and, afterwards, the sequential publishing of those information as "product" of the sensors the rover should have. The ROS2 architecture, has the goal of elaborating the data, yielding the hybrid traversability cost map relative to each sample. Those maps are then combined together to build an overall, "global", cost map updating each time the portion of the map involved. The update, in order to preserve as much data as possible from the previous samples, relieving the system from some additional computations, will focus only on filling out newly discovered cells due to the navigation and updating previously known cells **if and only if** the new cost results to be higher than the previous one, trying to follow always a cautious approach.

To verify the actual usability of the product of the ROS2 pipeline, at the end of the provided samples an additional node has been created to retrieve the complete map and apply a path planning algorithm on it, which should elaborate the optimal path based on the data provided by the traversability analysis.

Path planning

The algorithm chosen for these tests is one of the most widely employed planning algorithm, the $A*$ algorithm. Presented for the first time in [62], the $A*$ represents an enhanced version of Dijkstra's Algorithm, specifically tailored for single-destination optimization by incorporating elements of the Greedy Best First

Search. Unlike Dijkstra's algorithm, which determines the shortest-path tree from a given source to all potential destinations, the $A*$ focuses solely on identifying the shortest path between a designated source and target. It achieves this by prioritizing paths that appear to bring the search closer to the goal and is able to effectively deal with obstacles.

Figure 5.21: Image from [63] showing a comparison between the three algorithms.

It can be seen in the figure above, how the $A*$ is able to take the best from both its "components". It is capable of finding the correct path as the Dijkstra's algorithm, but also narrowing the amplitude of the map explored as the Greedy Best First Search does, providing the best trade off between the two.

5.3 Simulation tests

Three are the main concepts around which the simulation tests revolve: the overall map creation, the overall map update and the path planning algorithm results.
The map has been designed to be sufficiently large to store all the published cost maps and is made entirely of unknown cells, which will be filled out as the data come in. Being a global container, it is required that all the grid maps are represented with the same resolution. After the preliminary stages, a value of 10cm for the side of the cells has been chosen since its minimum value in all the samples fell behind this threshold. As for the previous cases, the map will be inserted into an *OccupancyGrid* message, sent to the node in charge of the path planning and retrieved from RViz displays. The zero of this overall map is set to coincide with the zero of the first sample considered, which will be the starting point of the motion of the rover. It will also be the origin of the fixed frame, *odom* visible in figure 5.22, first one of the three frames displayed in the visualization. The other two represent, respectively, the base-link, of the rover, following, for this reason, its motion and the camera mounted on it.
The next, and most important, aspect of these last tests, is the update of the map. Every time a new sample is acquired and processed, it must be used to update the content of the overall map. Before doing this, a premise on the motion of the rover must be considered. Being the cost map re-projected into a bi-dimensional plane, the slope of the navigated terrain must be accounted in the evaluation of the cells to move forward. What has been done is the computation, in the geometry-based node, where terrain slope is retrieved, of the average slope of the supposed step size between the current and the next sample. Once that value is obtained, the step size is multiplied by the cosine of that slope to return the projected step in the 2D grid map, needed for the updates. The value is sent to the *final_node* and employed to compute, each time, the starting point from where the updatable region begins. The criterion applied for the actual update operation, as described in section 5.2.2, touches only unknown cells, discovered by the current sample, and known cells which new cost is higher than the previous one.
The obtained result, is a map which grows every processed sample without losing its overall coherence, as can be seen in an example of what displayed in RViz after some iterations:

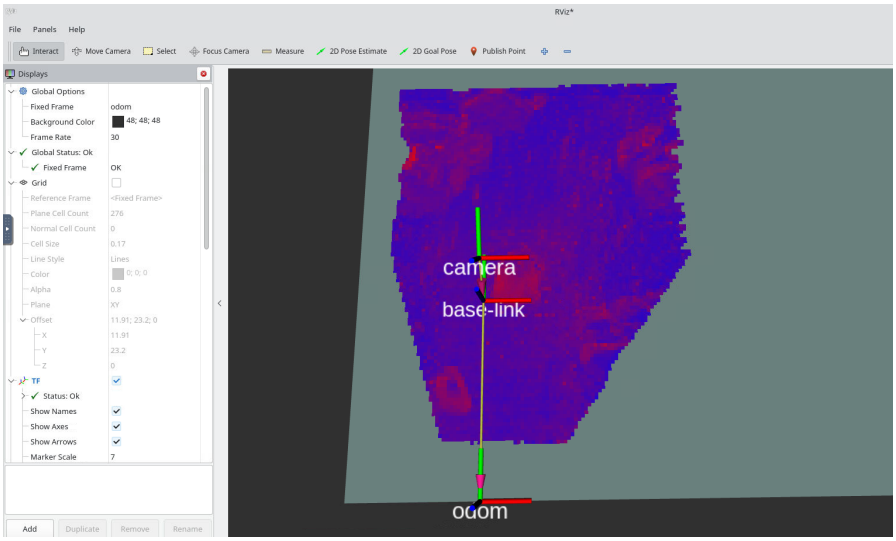

Figure 5.22: Updated overall map.

The final point to be covered is the planning algorithm. The additional node in charge of the planning operations is subscribed to a topic where an *OccupancyGrid* message is sent only when the complete map is available, since, currently, the only aim of its application is to show the possible usage of the produced data for autonomous navigation. Once the message arrives to the node, its content is extracted and the path planning A^* algorithm is applied to compute the optimal path between two tunable points of the map. The result of the algorithm is then processed to create a *Path* message, from the *nav_msgs* package, and sent out to be displayed on the visualized map through RViz.

5.4 Performances' evaluation

From the start of this thesis work, it has been stated that the final goal of the framework would have been an effective terrain traversability analysis of harsh and unstructured environments, to be employed in a navigation system ensuring the safety of the vehicle. From the results of the tests, carried out in the previous sections, it is possible to confirm that the desired result has been achieved successfully. The obtained grid map is the tangible proof of the assessment of terrain traversability as a result of a hybrid approach involving both semantic, physical and geometrical information.
Another important reached objective, to be pointed out, is the high level of detail in the result yielded by this framework, in comparison with other similar solutions, e.g. the hybrid architecture presented in [49].

Another side to be taken into account revolves around the execution time requested to get to the final result. Being the environment in exam free from dynamic obstacles, as could be other vehicles or pedestrians in a structured urban environment, in which the online performances play a key role, the time requested could be considered as less relevant, but cannot be totally ignored. From the measurements done on two different sets portraying two different simulated environments, the time needed from the acquisition of the images to the publishing of the final hybrid traversability cost map ranges between 2 to 6 seconds, as is defined more in detail in table 5.1 and can be visually seen in figure 5.23. It must be considered that the UGV on which the framework is implemented could heavily affect in both directions this estimate, depending on the hardware equipped. However, evaluating the available information from the simulation tests and considering the speed of the currently fastest Martian rover, Perseverance, being 0.161 km/h (or 0.1 mph), it can be considered as a completely reasonable time request.

Number of samples in the set	Min completion time [s]	Max completion time [s]	Avg completion time [s]
60	2.6	6.04	3.083
80	2.25	6.2	2.935

Table 5.1: Completion times

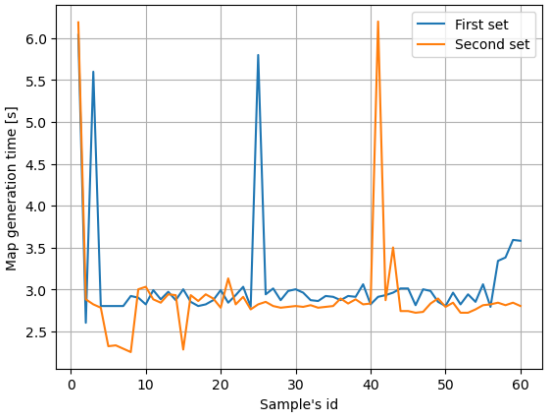

Figure 5.23: Behaviour of the completion time over the sets' samples. The second set has been cut to the 60^{th} sample to be compared with the first one

The last aspect to be discussed is the result of the path planning algorithm applied on the retrieved cost map from the framework. As expected, the planner is able to correctly plan the optimal path, from the starting point to the desired destination, following the best path relative to the costs displayed by the map. This behaviour ensures that the UGV navigates safely through the environment it is placed in and reaches the desired target point, avoiding possible obstacles or hazardous areas encountered throughout its navigation. In the figure below the success of the path planning can be seen, represented by the green line visible clearly on the traversability cost map:

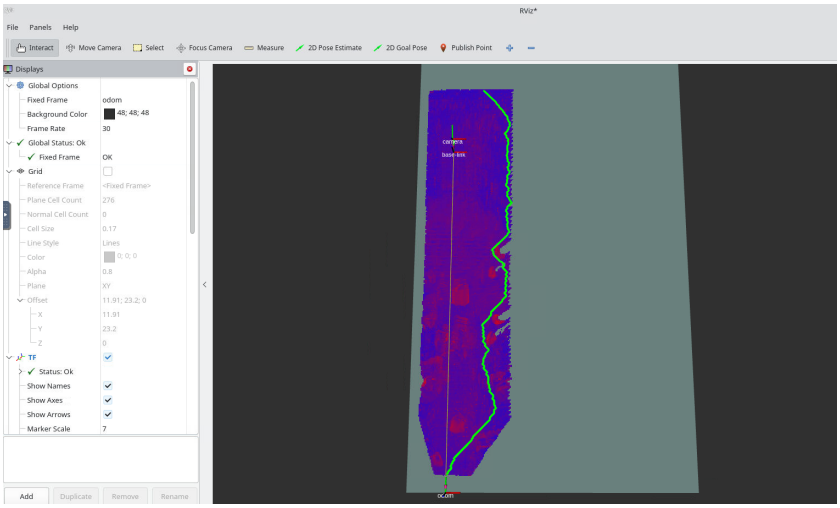

Figure 5.24: Display of the planned path using the hybrid traversability map.

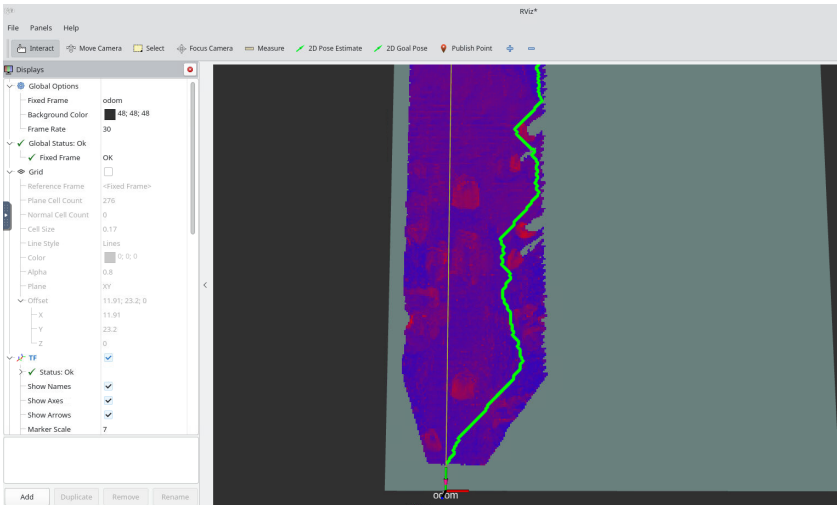

Figure 5.25: Close up of the planned path showing obstacle avoidance

Chapter 6

Conclusions and future improvements

6.1 Conclusions

In the first chapter, the problem of terrain traversability analysis in the context of planetary exploration was outlined and the proposal of a framework to handle this task, following a hybrid approach, was theorized. In the remaining chapters all the key steps of the development phases of the, aforementioned, architecture have been presented and detailed, from the theoretical research, studying the state of the art which lead to the definition of the two methods constituting the hybrid-approach, to the implementation and validation phases of the ROS2 package created. In conclusion is, therefore, possible to state that the presented framework, even if still highly improvable, is able to correctly operate a traversability analysis of the terrain surrounding the rover. The cost map, obtained as final product of the pipeline, as shown in section 5.3, contains all the necessary data to be processed by a path planning algorithm, like the one employed in the same section, fundamental in any autonomous navigation system. As introduced in chapter 1, and further explored in chapters 2 and 3, these information are essential in guaranteeing the safety of the vehicle during its navigation. With respect to the hybrid framework proposed in [49], a novel improvement has been presented, for what concerns the appearance-based side, enriching the level of detail in the first cost map by obtaining physical information about the terrain through roughness classification. This adjustment empowers the vehicle with the capability of evaluating from an additional point of view the environment, making even more robust the framework. This is achieved trying to decrease the possibility of complete failure in the traversability analysis by increasing the number of employed criteria without rising too much the required workload.

6.2 Future improvements

As mentioned in the section above, the framework, even if correctly working, could be highly improved. Being limited both in time and resources to spend on the project a list of all the considered adjustments and changes to the architecture, which could possibly further refine and improve its performances, is reported.
To have a more readable list, they will be divided into bullet points, each one dedicated to a section of the work:

- **Dataset**: For what concerns the dataset creation, many could be the improvements, the easier one being an increase in its size and variety, either by simply adding more images or by applying some data augmentation technique. The aim of this process would be an enhancement in the generalization capabilities of the networks, leading to more accurate predictions on unseen images. A more significant and challenging improvement, instead, could be represented by the creation of a hybrid real-synthetic set of images, to try to reach a point where the method, trained on this set, can be validated on real samples. It would address the problem of being forced to wait for a large and accurate set of real world images to employ the framework on a physical vehicle. The difficulty in this approach comes from two sides, the first one is the need of obtaining a more accurate simulated environment to gather the pictures from, finding the closest possible settings to the real Martian landscape. The second and more challenging aspect, is that to train the roughness classification networks at least the relative depth images are needed. This additional requisite, being non dependent on the developer represents an obstacle which can be overcome only by the publication of new data from the space agencies.

- **Appearance-based approach**: Similarly to the first point there could be a lot of further improvements, deriving from the exponential rate at which deep learning research progresses. A deeper investigation phase on the more suitable convolutional network could, in fact, reveal a different and more powerful network, with respect to UNet, e.g. Transformer networks. Another possible improvement, considered in the development of the framework, but never tried seriously, is the creation and employment of a dedicated multitask learning model. Similarly to the HydraNet used in the autonomous navigation system of Tesla vehicles [64], able to address simultaneously both semantic segmentation and roughness classification. As stated in [65], a properly designed and implemented multitask learning model presents numerous advantages like improved data efficiency, reduced overfitting through shared representations,

and fast learning by leveraging auxiliary information. Other than just joining together the tasks removing the need for two networks, it can potentially upgrade the learning capabilities of the appearance-based side obtaining better and more accurate predictions.

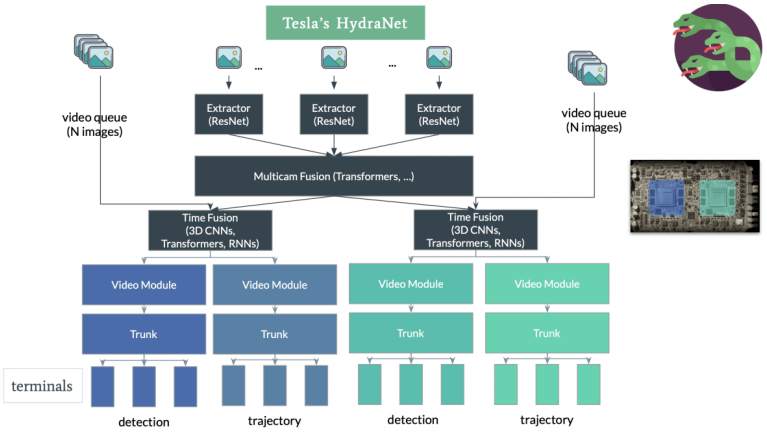

Figure 6.1: Tesla's HydraNet, a single network managing four different tasks[64].

- **Geometry-based approach**: The geometry-based side could benefit mainly from one adjustment. It consists in a switch of the approach as a whole from the mere slope evaluation, to a more refined, complex and high-performance method, like the one presented in [45], which was initially considered for this work. That approach, for example, would also yield better performances in the creation of a global map accounting for drift and uncertainties of the state estimation. As anticipated in section 4.2.3, the main issue regarding it revolved around the need of translating the available package into ROS2 nodes, being it created for ROS, but other adjustments would be required to adapt it from legged robots to wheeled rovers.
- **ROS2 architecture and simulation**: Regarding the ROS2 package, all adjustments will pertain to one main improvement in relation to the tests presented in section 5.3: evaluating the framework in a navigation simulator. This would require the use of the *nav2* package in charge of managing the autonomous navigation and, as starting point, the implementation of a simulated Martian soil to traverse in a software like Gazebo or Unity. Differently from what mirrored in this work, the vehicle would actually navigate through the environment using the data obtained from the framework as a way to plan the optimal path to follow with the aim of reaching some kind of target point. To have the system correctly functioning the blender file generated by oaisys must be employed as surface to traverse, maintaining, in this way, the correct terrain textures and objects the networks are trained to classify.

Bibliography

[1] Joshua Rapp Learn. *Why haven't humans reached Mars?* May 2023. URL: https://www.astronomy.com/science/why-havent-humans-reached-mars/ (cit. on p. 2).

[2] Mark Maimone, Andrew Johnson, Yang Cheng, Reg Willson, and Larry Matthies. «Autonomous Navigation Results from the Mars Exploration Rover (MER) Mission». In: *Experimental Robotics IX*. Ed. by Marcelo H. Ang and Oussama Khatib. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 3–13. ISBN: 978-3-540-33014-1 (cit. on p. 11).

[3] Arturo Rankin, Mark Maimone, Jeffrey Biesiadecki, Nikunj Patel, Dan Levine, and Olivier Toupet. «Driving Curiosity: Mars Rover Mobility Trends During the First Seven Years». In: *2020 IEEE Aerospace Conference*. 2020, pp. 1–19. DOI: 10.1109/AERO47225.2020.9172469 (cit. on p. 11).

[4] Michael McHenry et al. *MARS 2020 AUTONOMOUS ROVER NAVIGATION*. 2020. arXiv: 2003.05663 [cs.RO] (cit. on p. 11).

[5] Brandon Rothrock, Ryan Kennedy, Chris Cunningham, Jeremie Papon, Matthew Heverly, and Masahiro Ono. «SPOC: Deep Learning-based Terrain Classification for Mars Rover Missions». In: *AIAA SPACE 2016*. DOI: 10.2514/6.2016-5539. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2016-5539. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2016-5539 (cit. on pp. 11, 22, 35).

[6] K. McManamon, Richard Lancaster, and Nuno Silva. «EXOMARS ROVER VEHICLE PERCEPTION SYSTEM ARCHITECTURE AND TEST RESULTS». In: 2013. URL: https://api.semanticscholar.org/CorpusID:88504293 (cit. on p. 12).

[7] Dario Calogero Guastella and Giovanni Muscato. «Learning-Based Methods of Perception and Navigation for Ground Vehicles in Unstructured Environments: A Review». In: *Sensors* 21.1 (2021). ISSN: 1424-8220. DOI: 10.3390/s21010073. URL: https://www.mdpi.com/1424-8220/21/1/73 (cit. on p. 13).

[8] *The History of Artificial Intelligence*. August 28, 2017. URL: https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/ (cit. on p. 14).

[9] Eduardo Romera, José M. Álvarez, Luis M. Bergasa, and Roberto Arroyo. «ERFNet: Efficient Residual Factorized ConvNet for Real-Time Semantic Segmentation». In: *IEEE Transactions on Intelligent Transportation Systems* 19.1 (2018), pp. 263–272. DOI: 10.1109/TITS.2017.2750080 (cit. on p. 19).

[10] Vivekanandan Suryamurthy, Sushrutha Raghavan, Arturo Laurenzi, Nikos Tsagarakis, and Dimitrios Kanoulas. «Terrain Segmentation and Roughness Estimation using RGB Data: Path Planning Application on the CENTAURO Robot». In: Sept. 2019. DOI: 10.1109/Humanoids43949.2019.9035009 (cit. on pp. 20, 25, 28, 51, 52, 54, 55).

[11] Jiatian Wu. *Depth Detection with Vitis-AI on DPU Using Nod.AI Monocular Depth Detection Model*. https://www.xilinx.com/developer/articles/depth-detection-with-vitis-ai-on-dpu-using-nod-ai-monocular-dept.html. 2020 (cit. on p. 20).

[12] Luiz F. P. Oliveira, António P. Moreira, and Manuel F. Silva. «Advances in Forest Robotics: A State-of-the-Art Survey». In: *Robotics* 10.2 (2021). ISSN: 2218-6581. DOI: 10.3390/robotics10020053. URL: https://www.mdpi.com/2218-6581/10/2/53 (cit. on p. 21).

[13] Marek Pierzchała, Philippe Giguère, and Rasmus Astrup. «Mapping forests using an unmanned ground vehicle with 3D LiDAR and graph-SLAM». In: *Computers and Electronics in Agriculture* 145 (2018), pp. 217–225. ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2017.12.034. URL: https://www.sciencedirect.com/science/article/pii/S0168169917301631 (cit. on p. 21).

[14] Ash Vadolia. *Transforming Mining Operations: Harnessing UGV Robots for Innovation and Efficiency - case study news and blog posts*. Aug. 2023. URL: https://www.chironix.com/post/transforming-mining-for-innovation-and-efficiency (cit. on p. 22).

[15] Karsten Berns, Atabak Nezhadfard, Massimo Tosa, Haris Balta, and Geert De Cubber. «Unmanned Ground Robots for Rescue Tasks». In: *Search and Rescue Robotics*. Rijeka: IntechOpen, 2017. Chap. 4. DOI: 10.5772/intechopen.69491. URL: https://doi.org/10.5772/intechopen.69491 (cit. on p. 22).

[16] Yugang Liu and Goldie Nejat. «Robotic Urban Search and Rescue: A Survey from the Control Perspective». In: *Journal of Intelligent & Robotic Systems* 72.2 (Nov. 2013), pp. 147–165. ISSN: 1573-0409. DOI: 10.1007/s10846-013-9822-x. URL: https://doi.org/10.1007/s10846-013-9822-x (cit. on p. 22).

[17] Yan Guo, Aiguo Song, Jiatong Bao, Tang Hongru, and Jianwei Cui. «A Combination of Terrain Prediction and Correction for Search and Rescue Robot Autonomous Navigation». In: *International Journal of Advanced Robotic Systems* 6.3 (2009), p. 24. DOI: 10.5772/7229. eprint: https://doi.org/10.5772/7229. URL: https://doi.org/10.5772/7229 (cit. on p. 22).

[18] Andoni G. Moral Inza and Guillermo Lopez-Reyes. «Evolution of the Scientific Instrumentation for In Situ Mars Exploration». In: *Mars Exploration*. Ed. by Giuseppe Pezzella and Antonio Viviani. Rijeka: IntechOpen, 2020. Chap. 6. DOI: 10.5772/intechopen.93377. URL: https://doi.org/10.5772/intechopen.93377 (cit. on p. 22).

[19] Deegan Atha, R. Michael Swan, Annie Didier, Zaki Hasnain, and Masahiro Ono. «Multi-mission Terrain Classifier for Safe Rover Navigation and Automated Science». In: *2022 IEEE Aerospace Conference (AERO)*. 2022, pp. 1–13. DOI: 10.1109/AERO53065.2022.9843615 (cit. on p. 22).

[20] Panagiotis Papadakis. «Terrain traversability analysis methods for unmanned ground vehicles: A survey». In: *Engineering Applications of Artificial Intelligence* 26.4 (2013), pp. 1373–1385. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2013.01.006. URL: https://www.sciencedirect.com/science/article/pii/S095219761300016X (cit. on pp. 22, 24, 30).

[21] Mennatullah Siam, Mostafa Gamal, Moemen Abdel-Razek, Senthil Yogamani, Martin Jagersand, and Hong Zhang. «A Comparative Study of Real-Time Semantic Segmentation for Autonomous Driving». In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2018, pp. 700–70010. DOI: 10.1109/CVPRW.2018.00101 (cit. on p. 25).

[22] Yuanjun Zhu, Yunqiang Wang, Mingan Shao, and Robert Horton. «Estimating soil water content from surface digital image gray level measurements under visible spectrum». In: *Canadian Journal of Soil Science* 91.1 (2011), pp. 69–76. DOI: 10.4141/cjss10054. eprint: https://doi.org/10.4141/cjss10054. URL: https://doi.org/10.4141/cjss10054 (cit. on pp. 25, 28).

[23] Bir Bhanu, Peter Symosek, and Subhodev Das. «Analysis of terrain using multispectral images». In: *Pattern Recognition* 30.2 (1997), pp. 197–215. ISSN: 0031-3203. DOI: https://doi.org/10.1016/S0031-3203(95)00152-2. URL: https://www.sciencedirect.com/science/article/pii/S0031320395001522 (cit. on p. 25).

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. «U-Net: Convolutional Networks for Biomedical Image Segmentation». In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi. Cham:

Springer International Publishing, 2015, pp. 234–241. ISBN: 978-3-319-24574-4 (cit. on pp. 25, 26, 45).

[25] Dănuţ-Vasile Giurgi, Thomas Josso-Laurain, Maxime Devanne, and Jean-Philippe Lauffenburger. «Real-time road detection implementation of UNet architecture for autonomous driving». In: *2022 IEEE 14th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)*. 2022, pp. 1–5. DOI: 10.1109/IVMSP54334.2022.9816237 (cit. on p. 25).

[26] Dan C. Ciresan, Alessandro Giusti, Luca Maria Gambardella, and Jürgen Schmidhuber. «Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images». In: *Neural Information Processing Systems*. 2012. URL: https://api.semanticscholar.org/CorpusID:7725346 (cit. on p. 25).

[27] Jonathan Long, Evan Shelhamer, and Trevor Darrell. *Fully Convolutional Networks for Semantic Segmentation*. 2015. arXiv: 1411.4038 [cs.CV] (cit. on p. 26).

[28] Devin Schumacher. *Concatenated Skip Connection*. July 2023. URL: https://serp.ai/concatenated-skip-connection/ (cit. on p. 26).

[29] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. *Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs*. 2016. arXiv: 1412.7062 [cs.CV] (cit. on p. 27).

[30] Huikai Wu, Junge Zhang, Kaiqi Huang, Kongming Liang, and Yizhou Yu. *FastFCN: Rethinking Dilated Convolution in the Backbone for Semantic Segmentation*. 2019. arXiv: 1903.11816 [cs.CV] (cit. on p. 27).

[31] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. *Rethinking Atrous Convolution for Semantic Image Segmentation*. 2017. arXiv: 1706.05587 [cs.CV] (cit. on p. 27).

[32] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. *Segmenter: Transformer for Semantic Segmentation*. 2021. arXiv: 2105.05633 [cs.CV] (cit. on p. 27).

[33] Sitong Wu, Tianyi Wu, Fangjian Lin, Shengwei Tian, and Guodong Guo. *Fully Transformer Networks for Semantic Image Segmentation*. 2021. arXiv: 2106.04108 [cs.CV] (cit. on p. 27).

[34] Ruyi Zhou, Wenhao Feng, Huaiguang Yang, Haibo Gao, Nan Li, Zongquan Deng, and Liang Ding. *Predicting Terrain Mechanical Properties in Sight for Planetary Rovers with Semantic Clues*. 2020. arXiv: 2011.01872 [cs.RO] (cit. on pp. 27, 28).

[35] K. Iagnemma, Shinwoo Kang, H. Shibly, and S. Dubowsky. «Online terrain parameter estimation for wheeled mobile robots with application to planetary rovers». In: *IEEE Transactions on Robotics* 20.5 (2004), pp. 921–927. DOI: 10.1109/TRO.2004.829462 (cit. on p. 27).

[36] Yuankai Li, Liang Ding, Zhizhong Zheng, Qizhi Yang, Xingang Zhao, and Guangjun Liu. «A multi-mode real-time terrain parameter estimation method for wheeled motion control of mobile robots». In: *Mechanical Systems and Signal Processing* 104 (2018), pp. 758–775. ISSN: 0888-3270. DOI: https://doi.org/10.1016/j.ymssp.2017.11.038. URL: https://www.sciencedirect.com/science/article/pii/S0888327017306209 (cit. on p. 27).

[37] S. Chhaniyara, C. Brunskill, B. Yeomans, M.C. Matthews, C. Saaj, S. Ransom, and L. Richter. «Terrain trafficability analysis and soil mechanical property identification for planetary rovers: A survey». In: *Journal of Terramechanics* 49.2 (2012), pp. 115–128. ISSN: 0022-4898. DOI: https://doi.org/10.1016/j.jterra.2012.01.001. URL: https://www.sciencedirect.com/science/article/pii/S002248981200002X (cit. on p. 27).

[38] M B G Sakti, Komariah, D P Ariyanto, and Sumani. «Estimating soil moisture content using red-green-blue imagery from digital camera». In: *IOP Conference Series: Earth and Environmental Science* 200.1 (Nov. 2018), p. 012004. DOI: 10.1088/1755-1315/200/1/012004. URL: https://dx.doi.org/10.1088/1755-1315/200/1/012004 (cit. on p. 28).

[39] ANYbotics. *grid_map_rviz_plugin_example.png*. https://github.com/ANYbotics/grid_map/blob/master/grid_map_rviz_plugin/doc/grid_map_rviz_plugin_example.png (cit. on p. 30).

[40] Regis Hoffman and Eric Krotkov. «Terrain Roughness Measurement from Elevation Maps». In: *Proceedings of SPIE Mobile Robots IV*. Vol. 1195. Mar. 1990, pp. 104–114 (cit. on p. 30).

[41] D.K. Pai and L.-M. Reissell. «Multiresolution rough terrain motion planning». In: *IEEE Transactions on Robotics and Automation* 14.1 (1998), pp. 19–33. DOI: 10.1109/70.660835 (cit. on p. 30).

[42] Donald B. Gennery. «Traversability Analysis and Path Planning for a Planetary Rover». In: *Auton. Robots* 6.2 (1999), pp. 131–146 (cit. on p. 31).

[43] Dominik Joho, Cyrill Stachniss, Patrick Pfaff, and Wolfram Burgard. «Autonomous Exploration for 3D Map Learning». In: Jan. 2007, pp. 22–28. ISBN: 978-3-540-74763-5. DOI: 10.1007/978-3-540-74764-2_4 (cit. on p. 31).

[44] Péter Fankhauser, Michael Bloesch, Christian Gehring, Marco Hutter, and Roland Siegwart. «Robot-Centric Elevation Mapping with Uncertainty Estimates». In: *International Conference on Climbing and Walking Robots (CLAWAR)*. 2014 (cit. on p. 31).

[45] Péter Fankhauser, Michael Bloesch, and Marco Hutter. «Probabilistic Terrain Mapping for Mobile Robots with Uncertain Localization». In: *IEEE Robotics and Automation Letters (RA-L)* 3.4 (2018), pp. 3019–3026. DOI: 10.1109/LRA.2018.2849506 (cit. on pp. 31, 32, 61, 106).

[46] P. Bellutta, R. Manduchi, L. Matthies, K. Owens, and A. Rankin. «Terrain perception for DEMO III». In: *Proceedings of the IEEE Intelligent Vehicles Symposium 2000 (Cat. No.00TH8511)*. 2000, pp. 326–331. DOI: 10.1109/IVS.2000.898363 (cit. on p. 32).

[47] Roberto Manduchi, A. Castano, Ashit Talukder, and L. Matthies. «Obstacle Detection and Terrain Classification for Autonomous Off-Road Navigation». In: *Autonomous Robots* 18 (Jan. 2005), pp. 81–102. DOI: 10.1023/B:AURO.0000047286.62481.1d (cit. on p. 32).

[48] Daniel Fusaro, Emilio Olivastri, Daniele Evangelista, Pietro Iob, and Alberto Pretto. «An Hybrid Approach to Improve the Performance of Encoder-Decoder Architectures for Traversability Analysis in Urban Environments». In: *2022 IEEE Intelligent Vehicles Symposium (IV)*. 2022, pp. 1745–1750. DOI: 10.1109/IV51971.2022.9827248 (cit. on p. 32).

[49] Tiga Ho Yin Leung, Dmitry Ignatyev, and Argyrios Zolotas. «Hybrid Terrain Traversability Analysis in Off-road Environments». In: *2022 8th International Conference on Automation, Robotics and Applications (ICARA)*. 2022, pp. 50–56. DOI: 10.1109/ICARA55094.2022.9738557 (cit. on pp. 32, 33, 75, 101, 104).

[50] R. Michael Swan, Deegan Atha, Henry A. Leopold, Matthew Gildner, Stephanie Oij, Cindy Chiu, and Masahiro Ono. «AI4MARS: A Dataset for Terrain-Aware Autonomous Driving on Mars». In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2021, pp. 1982–1991. DOI: 10.1109/CVPRW53098.2021.00226 (cit. on pp. 34–36, 38).

[51] M. G. Müller, M. Durner, A. Gawel, W. Stürzl, R. Triebel, and R. Siegwart. «A Photorealistic Terrain Simulation Pipeline for Unstructured Outdoor Environments». In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 9765–9772. DOI: 10.1109/IROS51168.2021.9636644 (cit. on pp. 39, 41, 95).

[52] Washington University in St. Louis PDS Geosciences Node. *Analyst's notebook help*. URL: https://an.rsl.wustl.edu/help/Content/About%20the%20mission/MSL/Instruments/MSL%20Navcam.htm (cit. on p. 41).

[53] Adam Paszke et al. «PyTorch: An Imperative Style, High-Performance Deep Learning Library». In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf (cit. on p. 45).

[54] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG] (cit. on p. 47).

[55] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. *Open3D: A Modern Library for 3D Data Processing*. 2018. arXiv: 1801.09847 [cs.CV] (cit. on p. 51).

[56] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. *Caffe: Convolutional Architecture for Fast Feature Embedding*. 2014. arXiv: 1408.5093 [cs.CV] (cit. on p. 54).

[57] Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun. «Hands-On Bayesian Neural Networks—A Tutorial for Deep Learning Users». In: *IEEE Computational Intelligence Magazine* 17.2 (May 2022), pp. 29–48. ISSN: 1556-6048. DOI: 10.1109/mci.2022.3155327. URL: http://dx.doi.org/10.1109/MCI.2022.3155327 (cit. on p. 56).

[58] Weihuai Wu, Xiaomei Xie, Mingzhu Wei, Nian Liu, Xin Chen, Peng Yan, Omar Mechali, and Limei Xu. «Planetary Rover Path Planning Based on Improved A* Algorithm». In: Aug. 2019, pp. 341–353. ISBN: 978-3-030-27537-2. DOI: 10.1007/978-3-030-27538-9_29 (cit. on p. 61).

[59] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. «Generalized Intersection over Union». In: (June 2019) (cit. on p. 67).

[60] Robotic Operating System. URL: https://www.ros.org/blog/why-ros/ (cit. on p. 70).

[61] Massachusetts Institute of Technology. *Introduction to Ros*. URL: https://vnav.mit.edu/labs/lab2/ros101.html#:~:text=The%20ROS%20packages%20are%20the,item%20in%20the%20ROS%20software (cit. on p. 70).

[62] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. «A Formal Basis for the Heuristic Determination of Minimum Cost Paths». In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107. DOI: 10.1109/TSSC.1968.300136 (cit. on p. 97).

[63] *Red Blob Games: Introduction to A**. https://www.redblobgames.com/pathfinding/a-star/introduction.html (cit. on p. 98).

[64] Think Autonomous. *Tesla's HydraNet - How Tesla's Autopilot Works*. https://www.thinkautonomous.ai/blog/how-tesla-autopilot-works/. 2021 (cit. on pp. 105, 106).

[65] Michael Crawshaw. *Multi-Task Learning with Deep Neural Networks: A Survey*. 2020. arXiv: 2009.09796 [cs.LG]. URL: https://arxiv.org/abs/2009.09796 (cit. on p. 105).