POLITECNICO
MILANO 1863

Master of Science in Space Engineering

A Convolutional Neural Network Model as Image Processing in Cislunar Environment

Department of Aerospace Science and Technology

Advisor
Prof. Francesco Topputo

Coadvisor
Dr. Mattia Pugliatti

Student
Francesco Carrasso - 942144

Academic Year 2020-2021

Contents

List of Figures

List of Tables

Acknowledgements

In questi cinque anni, non ho soltanto avuto modo di viaggiare, e studiare in contesti diversi, ma ho avuto soprattutto la possibilità di condividere questo percorso con persone fantastiche, dalle quali ho imparato molto e senza le quali non sarei qui oggi con la determinazione e la sicurezza maturate. Un grazie infinito a chi c'è stato, da Torino a Milano e Copenhagen.
Ho imparato a scegliere ciò che è giusto, rispetto a ciò che è semplice. Il duro lavoro, i sacrifici e i momenti di debolezza sono stati la chiave per sviluppare le capacità apprese in questi incredibili cinque anni e per ottenere i risultati più ambiti.

Vorrei ringraziare innanzitutto il Prof. Francesco Topputo, per avermi dato la possibilità di lavorare su un argomento di mio interesse e per avermi dato fiducia, fondamentale per il successo di un lavoro. È stato un esempio di professionalità durante tutto il percorso. Ringrazio il Dr. Mattia Pugliatti, per avermi guidato in questi mesi, per avermi consigliato e per essere stato un esempio di determinazione, dedizione e serietà, elementi fondamentali per la crescita professionale. Grazie.

Ringrazio la mia famiglia, per l'appoggio e l'amore incondizionato che in questi cinque anni mi hanno guidato in ogni momento, lontano da casa, e per avermi sostenuto nelle mie scelte.

Ringrazio Anna Maria, per essermi stata affianco nelle scelte e nei momenti di crescita lungo tutti questi anni, sostenendomi in ogni istante, mostrando la versione migliore di me stesso.

Abstract

The last decade of technological developments saw the spread of Machine Learning techniques in numbers of different applications. Their enhancement, led by years of deep researches, allowed the building of high performance models empowering Artificial Intelligence systems. This allowed to automate operations that once were supervised by humans, with high accuracy. In Guidance Navigation and Control field, the chance to automate a process of navigation would let the satellite take a decision on its own in risk conditions, where an immediate operation is asked when communications are off, or distances are too high. The automation processes have been highly tested in the past years with standard Optical Navigation methods, relying on Image Processing techniques to perform state estimations of the satellites.
The high rate of growth of accuracy and complexity of Machine Learning methods, such as Neural Networks, provided to research community an alternative way to the standard techniques in OpNav. A way that must be properly studied and applied to space navigation to achieve a faster and reliable approach along with better performance than standard approaches provide.
The current work will carry out a comparison of a standard Image Processing techniques for Optical Navigation based on Center and Apparent Diameter (CAD) estimation of the Moon by making use of two algorithms: an ellipse fitting algorithm, and a Convolutional Neural Network that performs a regression task providing the center of mass and the radius of the Moon. The processes will be performed in the environment of the LUMIO mission. By making use of a rendering software, the images coherent with the mission will be generated and used for the comparison, in order to provide a test set, as similar as possible to the real mission scenario, and a training set of 34940 images for the deep learning method. As such, 8935 images of the Moon, rendered accordingly to the mission scenario will be used as object of comparison between the two algorithms. A final chapter will provide to the reader the results, highlighting the strength and weaknesses of the different methods.

Sommario

Gli ultimi decenni di sviluppo tecnologico sono stati caratterizzati da una notevole e rapida diffusione di tecniche di Machine Learning in differenti settori e applicazioni. Tale sviluppo, portato in auge da anni di meticolosa ricerca, ha permesso di definire modelli altamente performanti e sofisticati che hanno agito da base per sistemi di intelligenza artificiale. In particolare, nel settore della Guidance Navigation and Control, la possibilitá di automatizzare processi di navigazione permette a sistemi quali satelliti di agire autonomamente sulla base dei dati ricevuti dall'esterno, come immagini, in condizioni ad elevata criticità, quando è richiesta un'azione immediata in casi di failure dei sistemi di comunicazione o ad elevata distanza dalla terra. Tuttavia, i processi di automazione sono stati altamente studiati e testati negli anni con sistemi di Optical Navigation standard, sulla base di algoritmi di Image Processing per definire una stima della posizione del satellite stesso.
L'elevato tasso di crescita dell'accuratezza e complessità dei metodi di Machine Learning, come Reti Neurali, fornisce ai ricercatori una soluzione alternativa ai metodi standard di OpNav. Una via e una soluzione che deve essere accuratamente studiata e sperimentata nel settore della navigazione spaziale, con lo scopo di ottenere un metodo sì veloce, ma ancora più accurato delle tecniche standard.
Il presente lavoro ha lo scopo di presentare un confronto tra un metodo standard di Image Processing per navigazione ottica basato sulla stima del centro di massa e del raggio della Luna utilizzando un algoritmo di ellipse fitting, con una Rete Neurale Convoluzionale per un processo di regressione per il calcolo del centro di massa e del raggio. Gli approcci, verrano sperimentati e valutati nel contesto della missione LUMIO. A tal fine, utilizzando un software di rendering, le immagini relative alla missione stessa verranno generate e utilizzate come mezzo di confronto, per fornire sia un test set coerente con l'orbita, sia un training set di 34940 immagini per l'approccio di deep learning. Per tal motivo, 8935 immagini della luna verranno utilizzate e fornite come input ai due approcci. Un capitolo finale, fornirà al lettore i risultati, evidenziando i punti di forza e debolezza dei due metodi.

Chapter 1

Introduction

The space economy is living one the best moments since the first man on the Moon. The last decades saw an exponential growth of interest in space exploration, which provided the perfect environment for increasing businesses and prolific ground for innovations. The efforts that scientists made in developing technologies that allowed the first satellites to navigate in deep space, grew along with new softwares and algorithm techniques that spread in civil applications. The huge amount of tools in computer science, and the appealing growth rate of space economy, attracted numbers of companies that well established themselves in the market. Clear are the examples of SpaceX, Virgin Galactic and Blue Origin that are acting as pioneers in the private space exploration shifting to next level the technological environment and opening new scenarios for the space exploration.
The opening of the space market to privates, grew along with the interest of reducing the costs for space missions. Among the uncountable examples, there is the contribution given by *CubeSats*, small satellites cube fashioned introduced in 1999, that aimed to drastically reduce the building cost of satellites, allowing smaller companies and universities to work on their own solutions for space missions, democratizing, therefore, the access to space exploration.
Such a democratization of access to space is still growing but it has the right tools for an outstanding future. In such a challenging environment, men and women, passionate about space, started to develop their own ideas, founding new companies with innovative solutions for space systems, but also proposing new tools and techniques.

The design of a space mission requires a detailed and scrupulous understanding and evaluation of numbers of phenomena that may interact with the spacecraft. The success of the mission is therefore led by a high cooperation strategy between different teams of scientists. Among the parts that act a fundamental role in achieving the mission results, the *navigation* methods had important advancements throughout the past years. Spacecraft navigation comprises mainly three parts[1]: the *mission design* where a reference trajectory is designed, describing the planned flight path of the spacecraft; the *orbit determination* that keeps track of the actual position of the spacecraft while the mission is in flight; and the *flight path control* that creates maneuvers to bring back the spacecraft to the reference trajectory.

[1]https://solarsystem.nasa.gov/basics/chapter13-1/. Last accessed: 21-11-2021

The information and the measurement of the spacecraft state, defining its angular quantities (e.g. right ascension and declination) and its velocity have always been provided by the well established Deep Space Network (DSN), a worldwide network of spacecraft ground segment facilities. Through ranging pulse and the study of Doppler effect, it has been always possible to have therefore an estimation of the spacecraft's state. Along with this technique, spacecrafts were, and still are, also equipped with imaging instruments, used to observe the spacecraft's destination planet and to carry navigation maneuvers in the context of *Optical Navigation*.

Images of planets or bodies have been used for years in deep space navigation. Typically, the images are downlinked from the spacecraft and analysts on ground apply several image processing techniques to localize the observed object in each image that, combined with radiometric tracking, are input in a filter which provides the spacecraft state [1, 2]. On the basis of the results, the navigation solution is uplinked back to the spacecraft.
Methods such this have become reliable in the results throughout the years, providing precise solutions. Nevertheless, recent advancements in on-board computing capabilities and technological improvements, along with new and efficient image processing algorithms, may allow the spacecrafts to rely on methods for complete autonomous navigation.

While the mission plans for sending humans beyond low Earth orbit are proposed in an increasing number, the need to develop improved methods for autonomous navigation is becoming largely important and essential. Although the combination of radiometric and optical navigation data are well understood, solutions of such this nature for crewed missions in the future may be undesirable for several reasons. The lack of communications, due to a failure in the relative systems, may present a critic scenario in space navigation for both expert and civil passengers. If needed, the vehicle shall be capable to return back to Earth without any command from the ground or simply navigate guiding the passengers. Likewise, for robotic missions, flyby events at high speed, when the Sun is between the Earth and the spacecraft, may result in a drastic scenario. Consequently, the time delays that radio signals introduce in navigation in deep space present a relevant problem that needs to be solved, and full autonomous systems can provide a solution.

1.1 The state of the art

Historically, *OpNav* methods have been widely used since the early 70s, since when Mariner 9 [3] incorporated firstly *OpNav* measurements into mission orbit determination process [2]. In that case, the radiometric tools where combined with a limb detection algorithm to determine the relative location of the spacecraft with respect to Mars. In 1979, the Voyager 1 and Voyager 2 spacecrafts [4] used *OpNav* to meet the principal mission objectives, too. The mission around Jupyter relied on the use of the image as complementary tool for the corrupted data obtained from the radiometric measurements. The images of the planet's satellites, were obtained against a star background; the position of the satellite along with star locations, helped in retrieving the Jupiter-relative spacecraft orbit estimate by making use of filters, like the well proven and used Kalman filter.

In the 1990s, a particular interesting case was provided by the Galileo mission, that consisted of an orbiter and a probe that entered Jupiter's atmosphere. During the transit of Jupiter, in 1991, the deployment of the high-gain antenna failed, forcing Galileo to use its low-gain antenna for communication throughout the mission, reducing the communication bandwidth [2]. The reduction to 160 bps presented a problem for the *OpNav*, since the images needed to be downlinked to Earth for processing [5]. This forced to study and develop *OpNav* method that dealt with reduced data rate, providing advancements in image processing.
In 1998, the Deep Space 1 (DS1) mission was launched as part of the NASA New Millennium Program [6]. It introduced a new *AutoNav* system that was required to used optical images for asteroids for interplanetary orbit determination, to use the ion propulsion system to maintain and control the orbit and to provide the updates of the target position after flyby [2]. This approach was designed to make DS1 the first planetary mission capable of autonomously navigating all post-injection mission phases.

The reliable techniques developed for hundreds of missions, based on *limb fitting* or *star localization* have been used for recent missions. The success they had in the years and solutions they proposed, made them an essential approach to optical navigation and created with no doubt the basis for future advancements.
New Horizons OpNav was performed by processing images of the targets in front of a background field of catalogued stars [7]. Among the applied solutions, the star-based approach, allowed to compute the centers of the stars and minimize their differences in a least-squares sense against the predicted star centers provided by the catalogue. The inertial attitude of the camera was determined and along with the computed centers, passed to the orbit determination filter.

Other solutions, again, have been implemented for AIDA, the Asteroid Impact and Deflection Assessment mission, a joint mission between ESA and NASA for demonstrating the kinetic impactor technique for planetary defense [8]. The spacecraft HERA aims to be a highly autonomous system for space navigation, in a particularly challenging environment about a binary asteroid system. Here, the image processing techniques were developed on the basis of *centroiding* and *feature tracking* approaches, using the maximum correlation with a *Lambertian sphere* to determine the position of the asteroid in the field of view of the camera and features extraction, to estimate features between consecutive images to estimate the displacements of different relevant points [8].

The navigation systems based on *limb scanning* techniques spread with particular success since the first use in the Voyager mission [9, 1]. It worked by "*taking an assumed body center and then generating a handful of scan direction emanating from this center point*" [1]. In particular the length of the scan were based on the body's apparent size showed in the image. From each scan an intensity profile was generated. Then, it was compared to an expected intensity profile based on a relative state and body model, allowing to update iteratively the relative state, defining a more accurate state of the spacecraft.

When the spacecraft approaches the body target in short distances, however, solutions like *limb fitting* or *star horizon-based* become undoubtedly no more reliable. In this case, *feature tracking* may present a solid solution. The *Origins, Spectral Interpretation, Resource Identification, Security-Regolith Explorer* OSIRIS-REx spacecraft was launched on September 8, 2016 for the Bennu asteroid sample return mission. After reliability issues with hardware and test program of LIDAR system, an alternative backup solution to the LIDAR, based on Natural Feature Tracking (NFT) was introduced [10]. This optical navigation system compared observed images to a set of terrain models of the asteroid rendered in real-time from a catalog stored in memory. Among all the options for a backup solution to the LIDAR, the optical navigation, once again, was the most promising and mature technique to implement to achieve the Touch-and-Go requirements of the mission. The NFT system was designed to estimate the orbital state of the spacecraft by a matching process of the features. The images from the camera were matched against the predicted appearance of the features, rendered on-board. The approach predicted which features it was expected to be seen in the image, rendered the expected appearance of the feature, and matched the prediction to the actual image. The information of the location of the features were finally used to update the knowledge of the camera position and orientation by a Kalman filter.

However, the feature detection and matching algorithm, did not only spread in the space field. Along with these applications, more algorithms were introduced in robotic and civil applications enabling sophisticated vision-based relative navigation algorithms for robotic spacecrafts. The approaches introduced by ORB, SIFT and SURF methods resulted in being a viable solution for feature detection in SLAM algorithms. *Cho et al.* [11] proposed a relative navigation scheme in the space, based on these models. In particular, two different features detectors, the Shi-Tomasi corner detector and the SURF blob detector, followed by selection and filtering steps were used for feature detection over different scale of distances to the target. A robust-pattern matching method based on Gaussian Mixture Model was applied with the aim to enable target re-acquisition to provide fault-tolerant and automated relative navigation. The authors finally applied a fixed-lag smoothing SLAM algorithm to obtain pose, linear and angular velocity estimated with high accuracy.

The success *Machine Learning* had in the past years, made this new field an interesting solution for the image processing techniques in optical navigation. Among the methods that ML provides, one of the most interesting is *Deep Learning*, largely applied for image processing tasks. Taking advantage of the capabilities of *neural networks* to extract relations from images, uncountable are the solutions that space scientists have tried to apply for navigation purposes.
In [12, 13], a CNN with U-Net style, called LunaNet, was developed as image processing method. It visually detected craters in a simulated camera frame, and the detections were matched to known lunar craters in the region of the estimated spacecraft position. The model was developed on the idea of the already existing network DeepMoon [14], that applied a CNN to detect craters from a digital elevation map (DEM). To make the approach sensible to shadow variations, LunaNet used camera images to accommodate shadows or different forms of noises. The model output a gray-scale image with bright values indicating the pixels that were predicted to be part of crater rim. These outputs were then processed to fit an ellipse to the detected craters. Overall, the combination of LunaNet and an EKF produced interesting results in final position and velocity estimation error with respect to a standard image processing crater detection method.

On the basis of Natural Feature Tracking, *Campbell and Furfaro* [15] presented a CNN, that trained with a series of images rendered from digital terrain map (DTM) of lunar surface, returned the position of the observer relatively to the DTM. Here, the simplest problem of relative terrain navigation was considered, constraining the motion to one dimension. Taking advantage of the Lunar Reconnaissance Orbiter (LRO) digital terrain maps availability, a small 1024×1024 pixel section was selected to provide the raw data for image generation. Each pixel location along the direction of motion was treated as a class, providing 1024 classes. Each of the train image was labeled by where along this line the center lied. The images of 128×128 were sampled from the nadir base image every 8 pixel horizontally. Basically, the CNN received a 128×128 image labeled with a 1024 element one-hot vector that provided the center location. The image was then processed by three convolutional layers each followed by a max pooling layer to provide in the end the estimate center position in the image.

The flexibility of *neural networks* to deal with images, made their application not necessarily related to the picture of a planet or a moon. *Sharma and D'Amico* [16], proposed a neural network based method for on-board pose estimation of a known noncooperative spacecraft using monocular vision. This approach, used a CNN with three branches to solve the problem of relative attitude estimation. One branch performed a state-of-the-art object detection algorithm providing a bounding box around the target spacecraft. The region inside the bounding box was then input into the other two branches to determine the attitude by classifying the input region into discrete coarse attitude labels before regressing the a final estimate. The method then performed an estimation of the relative position by using the bounding box and the estimated relative attitude.

The state of the art the the optical navigation methods proposes is solid and well proven. Uncountable are the methods that had success in the past and acted as guideline for the future solutions. Among them, methods based on center and apparent diameter (CAD) finding, were deeply studied and proven through the years. An explanation of the analytic details behind a CAD method for both standard image processing and neural network will be presented later in dedicated chapters.

1.2 The case study: LUMIO mission

The Lunar Meteoroid Impact Observer (LUMIO) mission aims "*to observe, quantify, and characterize the meteoroid impacts by detecting their flashes on the lunar farside*"[2].
The spacecraft LUMIO is a 12U CubeSat with a mass of approximately 20kg, that is deployed into a quasi-polar selenocentric orbit. From a lunar high-inclination orbit, the spacecraft will autonomously determine its trajectory to reach a Moon-Earth L2 point, performing the cruise phase. When the lunar disk illumination will be less than 50%, it will perform autonomously the scientific task without direct coordination from the ground through the camera. Additionally, as a secondary application of the scientific investigation, the LUMIO-Cam will be

[2]https://directory.eoportal.org/web/eoportal/satellite-missions/content/-/article/lumio . Last access: 17-11-2021

also used to prove the autonomous navigation by making use of an horizon-based navigation algorithm, allowing to estimate the spacecraft state without relying on ground station tracking.

The scientific relevance Impacts due to Near Earth Objects (NEOs), may cause critical events on Earth, potentially leading to humanity extinction. Although the probabilities of these events happening are low, they require to be studied. In particular, meteoroids are small fragments of asteroids and comets with masses reaching 104 kg. Their detection may be hard, even with dedicated studies. Nevertheless, they may be observed indirectly by studying the impacts and the light flashes from the solid surfaces like the Moon's one.
The detection of the light flashes at the Moon, are typically observed by detecting a local spike of the luminous energy in the visible spectrum, pointing at the lunar night side. For this purpose, a dedicated camera LUMIO-Cam has been design to record the flashes. The camera consists of a 1024×1024 pixel detector, with pixel size of $13 \times 13 \, \mu m^2$, with a readout frequency of around 15 MHz. To take into account the maximum and the minimum ranges presented in Tab. 1.1, the optics behind this system has been designed to have a 6° FoV, a focal length of $127 \, mm$, and an aperture of $55 \, mm$.

To perform the scientific objective, the spacecraft will fly on a quasi-halo orbit around the L_2 point of the Earth-Moon system. Considering two consecutive orbits, LUMIO will perform engineering operations in one of them, dedicating the second orbit to science investigation.

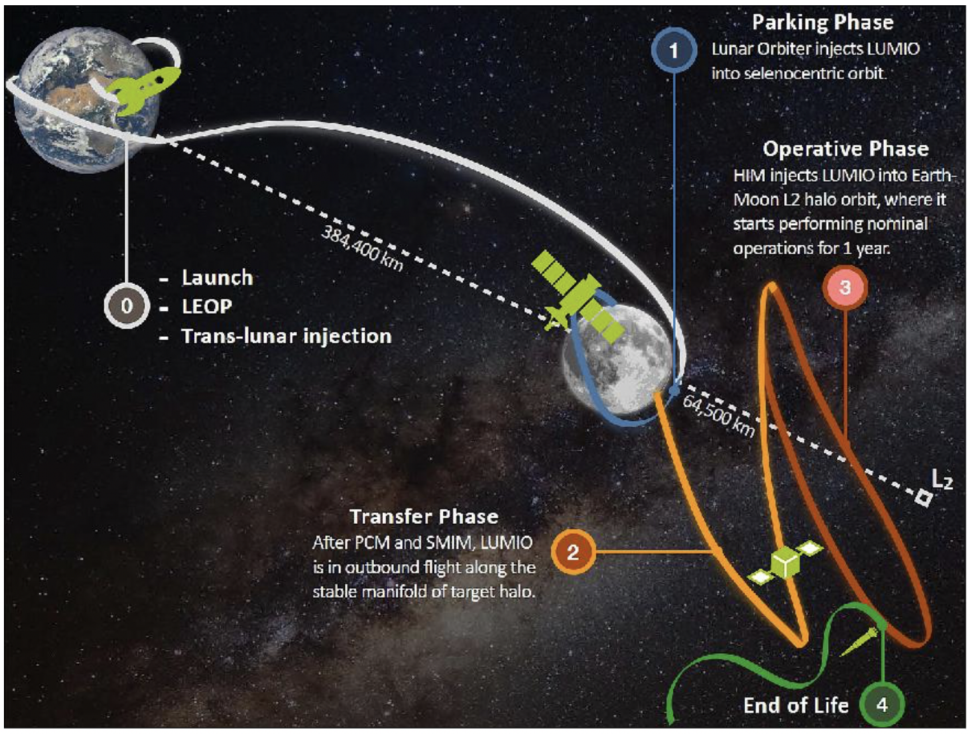

Parking Phase
Lunar Orbiter injects LUMIO into selenocentric orbit
1
Operative Phase
HIM injects LUMIO into Earth-Moon L2 halo orbit, where it starts performing nominal operations for 1 year.
3
384,400 km
0
- Launch
- LEOP
- Trans-lunar injection
64,500 km
L2
Transfer Phase
After PCM and SMIM, LUMIO is in outbound flight along the stable manifold of target halo.
2
End of Life
4

Figure 1.1: Sketch of LUMIO mission phases. Courtesy of [17].

Min range to moon	**Max range to moon**	**Halo period**	**Earth-moon synodic period**	**Jacobi constant**
35,525 km	86,551 km	14,74 days	29,84 days	3.09

Table 1.1: LUMIO halo orbit properties. Courtesy of [18].

The Optical Navigation System According to [18], in this case, a trade-off has been carried out between radiometric tracking, x-ray pulsar navigation, celestial triangulation, and the horizon-based navigation techniques. Undoubtedly, the objective of demonstrating the feasibility of autonomous on-board navigation, could not include the ground-based radiometric solutions. Considering the sensor miniaturization constraints for x-ray pulsar navigation and minor accuracy of celestial triangulation with an almost full field of view of LUMIO-cam, the horizon-based techniques turned out to be the best solution for the problem [18].

In particular, the navigation system was inspired by the methods proposed by *Christian et al.* in the *Noniterative Horizon-Based Optical Navigation by Cholesky Factorization* [19]. Proceeding with such a method, it is possible to estimate the camera-to-object relative position vector in the camera frame by relying on the estimation of the full ellipse of the moon, by ellipse fitting, and the retrieving of the range vector by a noniterative and novel approach presented by *Christian et al.* [19]. Once the range camera-to-moon is retrieved, an Extended Kalman Filter is applied to estimate the spacecraft state, that is the position along with the velocity.

The mission is one of the two winners of the ESA General Studies Program SysNova contest to design a CubeSat mission to the Moon. It has been proposed by a consortium composed by Politecnico di Milano, TU Delft, EPFL, S[&]T Norway, Leonardo S.p.A and the University of Arizona.

1.3 Research question

In the framework of the LUMIO mission, the objective of this work is to carry out a performance comparison between a standard CAD image processing based on ellipse fitting of the illuminated arc of the Moon, and a Neural Network architecture designed for a regression task, that provides the radius and the center of mass of the Moon. Therefore, the following main research question can be stated:

> "*Can a Convolutional Neural Network based architecture improve the accuracy of a standard CAD image processing technique?*"

As such, the main purpose is to understand how and if a neural network can improve the results of a well established standard CAD method for optical navigation. The capability of deep learning to retrieve relations between inputs and outputs, only on the basis of the pixel arrays of images, may outperform the weakness of standard image processing techniques when dealing with poorly illuminated images. Consequently, the sub-questions can be investigated:

> "*How do the different approaches deal with different illumination conditions of the Moon?*"

Regarding, instead, the optimization of the network, it must be capable to generalize the mapping between input and output:

> "*What kind of Neural Network architecture can lead to the best test error?*"

1.4 Description of the Document

The previous chapters have been used to introduce the reader to the environment of the navigation in space, presenting a descriptive survey of the main methods that have been used in the past and that have built the road for the future algorithms.

After having acquired a general picture of what has been done for space navigation, a more detailed description of the more used approaches will be presented to the reader. In the second chapter, a survey of different approaches used in *OpNav* is analyzed, presenting the setting of the reference frames for operations and a dedicated section for the *Image Processing* algorithm used for the standard CAD approach.
In the third chapter, the reader will be introduced to the innovative and appealing world of Artificial Intelligence, by presenting the Machine Learning techniques and some of the Supervised Learning methods, considering the supervised task of this work. A dedicated section will be presented for the Neural Networks and Residual Network on which the current thesis relies on.
The chapter four will explain how the environment work has been setup, from the need of generating a dataframe for the image generation and the relative labels, to the camera setting in the rendering software. The choices made to carry out the image generation and the algorithms of standard CAD and Convolutional Neural Networks will be introduced to the reader. The chapter will therefore explain how the algorithms have been applied to obtain the desired results.

In the end, a comparison of the results will be presented, showing how and when the algorithms performed well or not. The reader will approach finally to the last chapter, where the results will be summed up and the future works described.

Chapter 2

Optical Navigation

Optical Navigation is the use of imaging data to aid in spacecraft navigation. A *camera* takes picture of the nearby target body and the spacecraft's attitude determination system provides an estimation of the attitude of the camera during the exposure. Then an accurate measurement of the image coordinates of the body becomes a measurement of the inertial direction from the camera to the target [20]. Interplanetary robotic missions have relied on a combination of radiometric tracking and optical observations for navigation for years. However, to improve the estimate of the spacecraft state, relative to the planet, the optical observations were all processed on the ground.

Overall, the optical navigation methods may be summarized in three main problems [20]:

- Given the position and the velocity of the spacecraft and the target, the camera attitude, and the camera's optical properties, where should the image target appear withing a picture?
- Given a digital picture of stars/targets, what are the measured coordinates of these images?
- After the image processing is done, the obtained measurements must be fed into the orbit determination filtering process.

The main core of an *OpNav* system is therefore represented by *Image Processing*, an ensemble of techniques aiming to extract the coordinates of the pixels of an image of a target within a picture.

Why to use OpNav? Having an increasing number of missions planning to send human on missions beyond low Earth orbit, the idea of having methods for autonomous spacecraft navigation has become a requirement, and a compelling need. Although previous missions in the past relied on a combination of radiometric and optical measurements, typically fused on ground to estimate the state of the vehicle, today, for future crewed exploration missions, these kind of solution are undesirable. In the condition of a communication failure, it is vital that the vehicle be capable of autonomous navigation for the safety of the crew. For robotic missions, autonomous navigation may be necessary for high-speed flybys of a planet or for flybys occurring when the sun is between Earth and the planet, helping to reduce the tracking requirements on ground-based infrastructures. These objectives must be achieved at the minimum computational and time cost.

2.1 Reference frames

To properly apply the *image processing* technique to the *OpNav* system, the reference frames must be accurately defined to provide the fundamental tools for orbit and state estimation. One of the conventional celestial reference frames, later applied for the purpose of the work (section 4.1.1), is the *Body Fixed Reference Frame*. As the name explicates itself, it is defined such that it is fixed accordingly to a certain orientation of the body's surface and rotate and translate with the body itself. A slight variation is the *Principal Axis BFRF* (In Fig. 2.1 an explanatory picture from NAIF[1]), where the xyz axes of the system are aligned with the principal axes of the body.

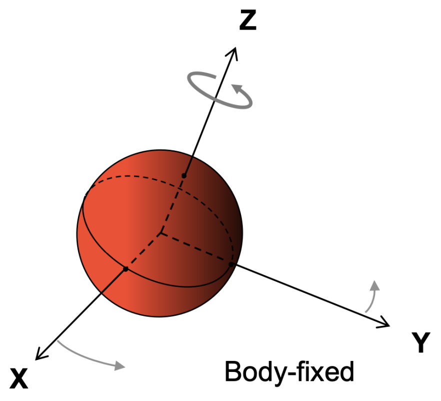


Figure 2.1: Body fixed reference frame.

2.1.1 Pinhole camera model

For a complete development of an image processing technique, the camera model must be defined. The simplest camera model available, and largely discussed in the literature [21], is the *pinhole camera model*, which describes the relationship of the projection of points in 3D space onto an image plane. The fundamental idea of this model is that the rays of light passing through the lens optical center are undistorted by the lens and continue until they intersect the focal plane array. The aperture of the camera model is approximated as an infinitesimal point, making the diffraction dominant and the purely refractive thin lens model not holding anymore[21].

To accomplish the *OpNav* task, here it must be defined the *camera reference frame*, centered in the pinhole point O and the *image plane reference frame*, which is placed along the z direction (corresponding to camera axis) on $z = f$ to avoid to reverse the image coordinates.

[1]https://naif.jpl.nasa.gov/pub/naif/toolkit_docs
/Tutorials/pdf/individual_docs/17_frames_and_coordinate_systems.pdf . Last accessed: 23-11-2021

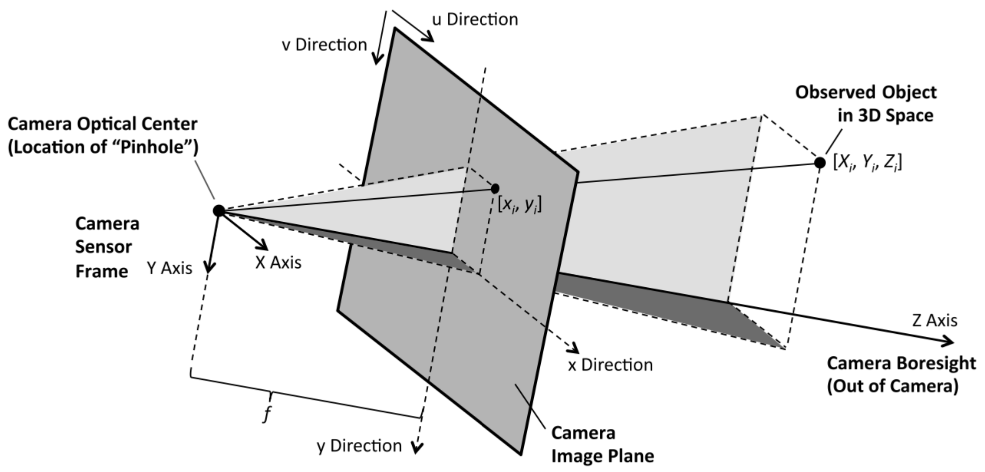


Figure 2.2: Basic pinhole camera model geometry. Courtesy of [22].

Therefore, given a generic point p of coordinates $\boldsymbol{X}=[X_p,\, Y_p,\, Z_p]$ relative to the camera reference frame, with the z being the optical axis, through triangles similarity it is possible to recover the p projection on the image plane through the mapping:

$$\begin{bmatrix} x \\ y \\ f \end{bmatrix} = \frac{f}{Z_p} \begin{bmatrix} X_p \\ Y_p \\ Z_p \end{bmatrix} \tag{2.1.1.1}$$

The coordinates will be then transformed to be expressed in the *Image Plane Coordinate Frame* in *uv* axes (Fig. 2.2) by making use of the camera calibration matrix accordingly to the properties of the system.

2.2 Standard Optical Navigation techniques: an Overview

The need to solve the issues due to latency of telecommunications signals for the spacecraft maneuvers, has been successfully satisfied in past years with different space proven techniques for *optical navigation.* Their application can vary depending of the phase of the mission. This translates in applying a method accordingly to how large is the image signal that is resolved in terms of amount of pixels.
By making use of the pixels of interest in the image plane, and by extracting features from the picture itself, it has been possible to retrieve the state of the spacecraft by using different tools like filters.

2.2.1 Moment algorithm or Center of Brightness

Moment algorithm is the the most primitive of the centerfindng techniques [20], based on the computation of the center of brightness of an array[2]:

$$s_c = \frac{\sum_i \sum_j iDN_{i,j}}{\sum_i \sum_j jDN_{i,j}}; \tag{2.2.1.1}$$

$$l_c = \frac{\sum_i \sum_j jDN_{i,j}}{\sum_i \sum_j jDN_{i,j}}; \tag{2.2.1.2}$$

with $DN_{i,j}$ the intensity value of the pixel in (*i,j*). This method, however, must be used carefully, considering particular image fixes. Indeed, the presence of non zero background values will move the computed (s_c, l_c) to move toward the center of the *DN* array. A filter must be therefore applied to deal with noises due to stars or lenses imperfections. A common procedure may also be to exclude from the sums all the pixels whose DN value is less than some minimum percentage of the brightest pixel in the image [20].

2.2.2 Analytic Function Fitting

Analytic Function Fitting was developed as a way to estimate the center of bodies in the image which are less than a couple of pixel across. Here, a *point spread function* (PSF) is fit to the DN values of the image to identify the center of the point source in the image [20]. A fit is carried out, usually with a *linear least-squares* estimation or *iterative least-squares estimation*, where the point spread function is such that to minimize the residuals between the predicted DN from the PSF and their actual values in the image. A typical example of PSF model is the 2D Gaussian function or the Lorentzian one. Any function capable to describe how a point source is blurred by the lens of the camera can be applied.
Using a 2D Gaussian, following the steps in [20], the brightness function becomes:

$$\begin{aligned} B(s,l) &= \frac{h}{2\pi} \exp\left(-\frac{(s-s_c)^2 + (l-l_c)^2}{2\sigma^2}\right) + b \\ &\equiv h\, N\left(\frac{s-s_c}{\sigma}\right) N\left(\frac{l-l_c}{\sigma}\right) + b \\ &\equiv h\, N(\xi(s))N(\eta(l)) + b \end{aligned} \tag{2.2.2.1}$$

where (s_c, l_c) are the coordinates of the peak of the Gaussian, σ is the standard deviation in pixels, h is the amplitude in DN, and b the constant background in DN. Then, *N(z)* is the normal probability distribution function, with zero mean and unit standard deviation; $\xi(s)$ and $\eta(l)$ convert s and l into units of standard deviations away from the mean.
In addition, the modeled DN is the integral of the brightness function is Eq. (2.2.2.1),

$$DN(s,l) = \int_{x=s-1/2}^{s+1/2} \int_{y=l-1/2}^{l+1/2} B(x,y)\, dy\, dx \tag{2.2.2.2}$$

[2]An image consists of tiny areas, arranged in regular rows and arrays, called pixels. Each pixel has coordinate values and a digital number *DN* that records the intensity of electromagnetic energy measured for the ground resolution cell represented by that pixel.

Being the fitting function not linear, the process is iterative. Therefore, considering the model parameters as $\{s_c, l_c, h, b, \sigma\}$, beginning with their *a priori* values computed from the image, the Eq. (2.2.2.2) is used to compute the expected DN values in each pixel. The spatial derivatives of DN(s,l) are the calculated with respect to the solution parameters and the residual are formed to construct each equation of condition. Data weight are applied according to the expected noise in DN, and results are fed into a least square algorithm: the process is iterated until convergence is achieved.

2.2.3 Normalized Cross-Correlation

In *Normalized Cross-Correlation*, the center of figure of the body in the image is found by correlating a template[3] of the predicted brightness values with the actual image. The process of correlation is performed as

$$\rho(\Delta s, \Delta l) = \sum_i \sum_j \frac{(DN_{i,j} - \bar{DN})(T_{i+\Delta s, j+\Delta l} - \bar{T})}{n\sigma_{DN}\sigma_T} \tag{2.2.3.1}$$

with T the template, $\bar{DN}$ is the average of the DN values over the considered pixels, $\bar{T}$ the average of the template values over the pixels and the σ the standard deviations of the DN from the image and the template over the pixels. The correlation is carried out for a number of $(\Delta s, \Delta l)$ and results in a correlation surface, the peak of which corresponds to the location of the center of the body.

2.3 Centroid and Apparent Diameter Optical Navigation

When satellite is approaching a moon or a planet, not only landmark navigation may be applied, particularly when the field of view of the camera is not completely saturated. Manifold are the methods that rely on the horizon. *Centroid and apparent diameter* (CAD) methods tend to be simpler in implementation since they do not require particular or specific landmarks to be found and matched to a catalog.
If the sizes of the body are completely known, then its apparent dimensions inside the image are related to the distance between the camera and the planet, accordingly to what explained in section 2.1.1, for the pinhole model. Moreover, if the center of the planet is also found in the image, the line-of-sight (LOS) direction from the camera to the planet is retrievable, too. Overall, this can easily provide a good estimation of the camera location.
The projection of a smooth ellipsoidal planet or moon on the image plane will be typically an horizon arc ellipse fashioned. Other sections are possible (e.g. hyperbolic or parabolic) if the satellites is at low altitudes, where horizon based OpNavs are not viable anymore. In any case, the observed horizon in the image will be obtained by slicing a cone with the image plane and solving a conic section problem for the horizon finding.

[3]The template used in cross-correlation is generated by rendering the view of the body using *a priori* knowledge of the scene.

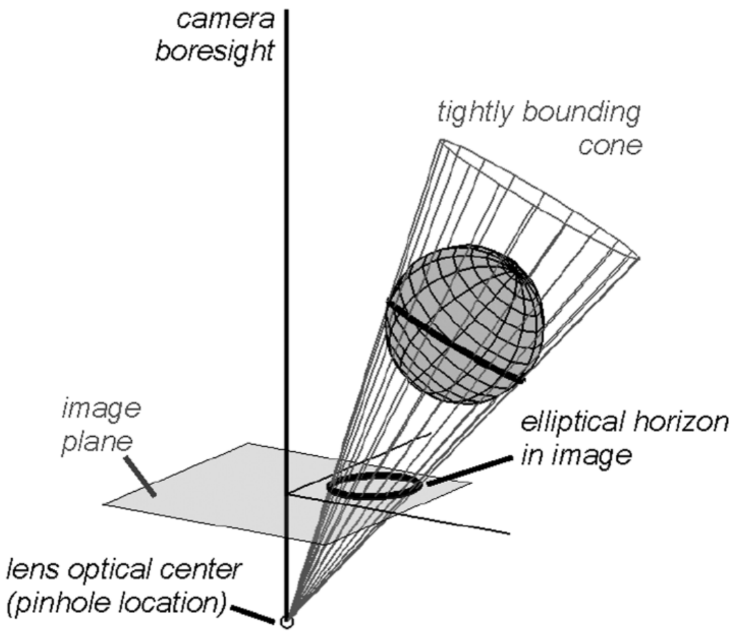

camera boresight
tightly bounding cone
image plane
elliptical horizon in image
lens optical center (pinhole location)

Figure 2.3: The cone tightly bounding the planet is sliced by the image plane, resulting in an ellipse arc. Courtesy of [1].

2.3.1 Geometry of planet observations

The shape of numbers of planets/moons of interest for OpNav can be easily and well approximated by a triaxial ellipsoid. Therefore, the surface of the modeled body follows the constraint

$$\mathbf{p_p}^T \mathbf{A_p} \mathbf{p_p} = 1 \tag{2.3.1.1}$$

that is the vector form of the implicit equation of a triaxial ellipsoid, with $\mathbf{p}$ a 3×1 vector describing a point on the body's surface with respect to the body's principal axis frame[4], and $\mathbf{A_p}$ a 3×3 symmetric positive definite matrix that describes the shape of the body

$$\mathbf{A_p} = \begin{bmatrix} 1/a^2 & 0 & 0 \\ 0 & 1/b^2 & 0 \\ 0 & 0 & 1/c^2 \end{bmatrix} \tag{2.3.1.2}$$

with $\{a,\ b,\ b\}$ the lengths of the planet's principal axis. To accomplish the tasks of the OpNav, it is preferable to express all the data in terms of the coordinate system aligned with the camera frame. Therefore, defining $\mathbf{T_p^C}$ the rotation matrix from the camera frame to the planet's principal axis frame and applying the transformation such that $\mathbf{p_p} = \mathbf{T_p^C p_C}$, and letting $\mathbf{A_C} = \mathbf{T_C^p A_p T_p^C}$ it is possible to recover

$$\mathbf{p_C^T A_C p_C} = 1 \tag{2.3.1.3}$$

[4]The p subscript defines a term in function of the principal axis frame of the body.

where $\mathbf{A_C}$ is the 3×3 symmetric positive definite matrix describing the planet's shape but in the camera frame. In this way, the points in the image that belong to the apparent horizon will correspond to the rays that start at the camera frame origin and go out tangent to the ellipsoidal planet's surface.
If the camera views an ellipsoidal planet form a relative position of $\mathbf{r}$, expressed in the camera frame, any ray $\mathbf{s_i}$ belonging to the cone that tightly bounds the triaxial ellipsoid, obeys to the constraint:

$$\mathbf{s_i^T}[\mathbf{Arr^TA} - (\mathbf{r^TAr} - 1)\mathbf{A}]\mathbf{s_i} = 0 \tag{2.3.1.4}$$

or simply

$$\mathbf{s_i^TMs_i} = 0 \tag{2.3.1.5}$$

with $\boldsymbol{M}$ a symmetric matrix of full rank given by

$$\mathbf{M} = \mathbf{Arr^TA} - (\mathbf{r^TAr} - 1)\mathbf{A} \tag{2.3.1.6}$$

Choosing a vector $\boldsymbol{s_i^T} = [x_i \; y_i \; 1]$ and substituting it into Eq. (2.3.1.5), what is obtained is simply the implicit equation for any conic section.

2.3.2 Ellipse Fitting

Given all the parameters of the camera involved for a mission, and considering the relations for pinhole model section 2.1.1, it is always possible to recover the body's size in the image plane, if the real dimension of the planet are known. In particular, the most fundamental task is to reconstruct the hypothetical ellipse that best fits the arc of the horizon detected from the body's image. This is more feasible the more the planet's shape is regular (e.g. spherical or ellipsoidal fashioned).

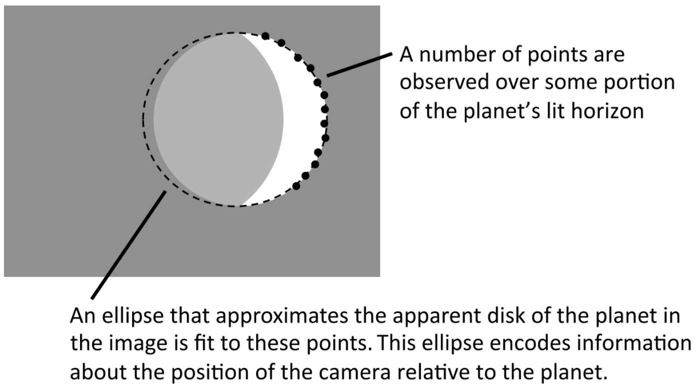

A number of points are observed over some portion of the planet's lit horizon
An ellipse that approximates the apparent disk of the planet in the image is fit to these points. This ellipse encodes information about the position of the camera relative to the planet.

Figure 2.4: A straightforward example of an ellipse fitting on the image of a planet/moon. Courtesy of [22].

Specifically, any conic section can be represented by an implicit quadratic equation:

$$F(x,y) = Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0 \tag{2.3.2.1}$$

where a point $(x\,,y)$ lies on the conic section. If $B^2 - 4AC < 0$ is satisfied, the equation will provide an ellipse. If all these parameters are known, it is possible to retrieve the ellipse coordinates center (x_0, y_0) in the image plane and the values of the semi-axes $\{a, b\}$. These values are easily achievable from classical geometry, and given by

$$x_0 = \frac{2CD - BE}{B^2 - 4AC} \qquad y_0 = \frac{2AE - BD}{B^2 - 4AC} \tag{2.3.2.2}$$

$$a = \sqrt{\frac{2[AE^2 + CD^2 - BDE + F(B^2 - 4AC)]}{(B^2 - 4AC)[\sqrt{(A-C)^2 + B^2} - A - C]}} \tag{2.3.2.3}$$

$$b = \sqrt{\frac{2[AE^2 + CD^2 - BDE + F(B^2 - 4AC)]}{(B^2 - 4AC)[-\sqrt{(A-C)^2 + B^2} - A - C]}} \tag{2.3.2.4}$$

For locating the planet or the moon in the image, the image must be searched for an ellipse. Once the candidate edge points are found, the ellipse must be fit to this data set.

Basic Ellipse fitting with Direct Least-Squares Estimation

One of the most efficient approaches for ellipse fitting is described by the direct least-squares algorithm introduced by Fitzgibbon, Pilu, and Fisher [23]. The first advantages with respect to other more robust noise techniques, are the less amount of memory required to carry out the operations, along with less time-consuming computations.
Recalling the ellipse in the form of Eq. (2.3.2.1), in the matrix form, it becomes

$$F(\mathbf{a}, \mathbf{x_i}) = \mathbf{a}\,\mathbf{x_i} = 0 \tag{2.3.2.5}$$

where (x_i, y_i) is a point of the conic section, $\mathbf{a} = [A\ B\ C\ D\ E\ F]^T$ and $\mathbf{x_i} = [x_i^2\ x_iy_i\ y_i^2\ x_i\ y_i\ 1]^T$. The form in Eq. (2.3.2.5) allows to arbitrarily rescale the the constants to transform the inequality constraint in an equality one

$$4AC - B^2 = 1 \tag{2.3.2.6}$$

However, due to noises, a point may not lie exactly on the ellipse, leading to $F(\mathbf{a}, \mathbf{x_i}) \neq 0$. Therefore, *Fitzgibbon et al.*, proposed an optimization problem based on the square of the model fit residuals in algebraic distance [23]:

$$min\ J = \sum_{i_1}^{n} [F(\mathbf{a}, \mathbf{x_i})]^2 = \mathbf{a}^T\mathbf{D}^T\mathbf{D}\mathbf{a} \tag{2.3.2.7}$$

with $\mathbf{D} = [\mathbf{x_1}\ \mathbf{x_2} \dots\ \mathbf{x_i}]$.
Additionally, if the equality constraint is rewritten in matrix shape $4AC - B^2 = \mathbf{a}^T\mathbf{C}\mathbf{a} = 1$ and adjoined with Eq. (2.3.2.7) with a Lagrange multiplier, then

$$min\ J = \mathbf{a}^T\mathbf{D}^T\mathbf{D}\mathbf{a} + \lambda(1 - \mathbf{a}^T\mathbf{C}\mathbf{a}) \tag{2.3.2.8}$$

The solution to the optimization problem is therefore a rank-deficient generalized eigenvalue problem

$$(\mathbf{D}^T\mathbf{D})\mathbf{a} = \lambda\mathbf{C}\mathbf{a} \tag{2.3.2.9}$$

Substituting the solution into the objective function

$$J = \mathbf{a}^T\mathbf{D}^T\mathbf{D}\mathbf{a} = \lambda\mathbf{a}^T\mathbf{C}\mathbf{a} = \lambda \tag{2.3.2.10}$$

leads to the optimal solution at the minimal positive eigenvalue.

Improved Ellipse Fitting with Direct Least-Squares Estimation

The Fitzgibbon's method, however, suffers from some practical difficulties [2]. Solving the eigenvalue problem previously presented in Eq. (2.3.2.9) leads to several issues with numerical instability. In addition, it works only on noisy data. Therefore, given a point lying exactly on an ellipse, the matrix $\mathbf{D}^T\mathbf{D}$ becomes singular and no solution is obtained. In the 1998, *Halir and Flusser* [24], presented a different approach overcoming the issues from the previous method. The structure of $\mathbf{C}$ and $\mathbf{D}$ was modified to simplify the eigenvalue problem. The new set of matrices were therefore defined as

$$\mathbf{a} = \begin{bmatrix}\mathbf{a_1}\\ \mathbf{a_2}\end{bmatrix} \qquad \mathbf{a_1} = \begin{bmatrix}A\\ B\\ C\end{bmatrix} \qquad \mathbf{a_2} = \begin{bmatrix}D\\ E\\ F\end{bmatrix} \tag{2.3.2.11}$$

$$\mathbf{C} = \begin{bmatrix}\mathbf{C_1} & \mathbf{0_{3\times3}}\\ \mathbf{0_{3\times3}} & \mathbf{0_{3\times3}}\end{bmatrix} \qquad \mathbf{C_1} = \begin{bmatrix}0 & 0 & 2\\ 0 & -1 & 0\\ 2 & 0 & 0\end{bmatrix} \tag{2.3.2.12}$$

$$\mathbf{D} = \begin{bmatrix}\mathbf{D_1} & \mathbf{D_2}\end{bmatrix} \tag{2.3.2.13}$$

$$\mathbf{D_1} = \begin{bmatrix}x_1^2 & x_1y_1 & y_1^2\\ x_2^2 & x_2y_2 & y_2^2\\ \vdots & \vdots & \vdots\\ x_n^2 & x_ny_n & y_n^2\end{bmatrix} \qquad \mathbf{D_2} = \begin{bmatrix}x_1 & y_1 & 1\\ x_2 & y_2 & 1\\ \vdots & \vdots & \vdots\\ x_n & y_n & 1\end{bmatrix} \tag{2.3.2.14}$$

Moving to a more compact notation, the scatter matrix is defined as $\mathbf{S} = \mathbf{D}^T\mathbf{D}$ with

$$\mathbf{S} = \begin{bmatrix}\mathbf{S_1} & \mathbf{S_2}\\ \mathbf{S_2}^T & \mathbf{S_3}\end{bmatrix} \tag{2.3.2.15}$$

$$\mathbf{S_1} = \mathbf{D_1}^T\mathbf{D_1} \qquad \mathbf{S_2} = \mathbf{D_1}^T\mathbf{D_2} \qquad \mathbf{S_3} = \mathbf{D_2}^T\mathbf{D_2} \tag{2.3.2.16}$$

If these partitioned matrices are inserted into the equations obtained in Eq. (2.3.2.9), the solution reduces to two equations

$$\mathbf{S_1}\mathbf{a_1} + \mathbf{S_2}\mathbf{a_2} = \lambda\mathbf{C_1}\mathbf{a_1} \tag{2.3.2.17}$$

$$\mathbf{S_2}^T\mathbf{a_1} + \mathbf{S_3}\mathbf{a_2} = \mathbf{0_{3\times1}} \tag{2.3.2.18}$$

Being the $\mathbf{S_3}$ generally invertible (it is singular when the points lie on a line, when no ellipse can be fitted), $\mathbf{a_2}$ is found as

$$\mathbf{a_2} = -\mathbf{S_3}^{-1}\mathbf{S_2}^T\mathbf{a_1} \tag{2.3.2.19}$$

Moreover, inserting Eq. (2.3.2.19) into Eq. (2.3.2.17), $\mathbf{a_1}$ is found solving the eigenvalue problem

$$\mathbf{M}\mathbf{a_1} = \lambda\mathbf{a_1} \tag{2.3.2.20}$$

with

$$\mathbf{M} = \mathbf{C}^{-1}[\mathbf{S_1} - \mathbf{S_2}\mathbf{S_3}^{-1}\mathbf{S_2}^T] \tag{2.3.2.21}$$

According to [24], even though Eq. (2.3.2.20) admits three possible solutions, only one elliptical solution is possible, and this is found for the eigenvalue that allows the constrain $4AC - B^2 > 0$ to be satisfied. This eigenvalue is the only solution for the eigenvector $\mathbf{a_1}$. Following such a procedure, all the parameters of the ellipse can be determined on the basis of the detected horizon points. Given the parameters, the estimated position of the center of body and its average diameter can be retrieved from the equations shown in Eq. (2.3.2.2).

This technique provides an efficient and accurate model for ellipse fitting. In the case noisy values were in the images, robust model fitting may be applied, like RANSAC models or its variation of MSAC, as presented in [2], largely studied in the computer vision field.

By making use of the geometric formulation shown in section 2.3.1 and the ellipse fitting methods, several ways have been developed to determine the relative position vector $\mathbf{r}$ of the body with respect to the camera frame, like the novel method of *Noniterative Horizon-Based Optical Navigation by Cholesky Factorization* that makes use of Cholesky factorization to map any shape into a sphere, simplifying the problem and recovering the range vector by a noniterative approach [19].

Chapter 3

Artificial Intelligence

First approaches to *Artificial Intelligence* in computer science come from the 1950s, when some pioneers of technological renaissance started asking if computers could be made to "think" [25]. Today, many and controversial are the definitions of AI, but they may mostly converge in defining it as *the effort to automate intellectual tasks normally performed by humans* [25]. AI is a field in the computer science that involves *Machine Learning* and *Deep Learning* techniques. In its initial formulation, it included approaches that did not involve learning at all: in chess programs, hardcoded rules were defined by programmers, trying to reach a sufficiently large set of explicit rules for manipulating knowledge. These led to the idea of *symbolic AI*.
The development of new technologies, that improved the computational power of processing units, allowed researchers to figure out new rules capable to solve even more complex problems such as image classification, speech recognition, object detection, and more. There, *machine learning* took *symbolic AI*'s place.

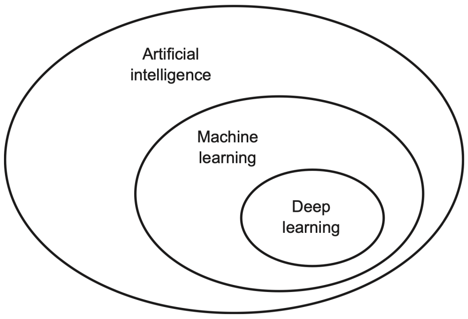


Figure 3.1: Artificial Intelligence structure.

3.1 Machine Learning techniques

More than 65 years ago, Alan Turing, in the famous essay *Computing machinery and intelligence* [26], asked if a machine could do the same things humans could do. Turing proposed that instead of writing programs that behaved like a human from scratch, it was needed a computer which *learned from past experience.* A machine capable to translate between two languages, will do so by learning from past examples and not from hardcoded

rules [25]. This means that a computer should perform a specific task by learning rules from data observation: that is *Machine Learning*.
The goal of machine learning is therefore to use past experience to learn how to accomplish a certain task such that the learned ability generalizes to future similar situations. This changes the paradigm of programming, as explained by Francois Chollet in [25], where the outputs of the process are no more the *answers*, but the *rules*.

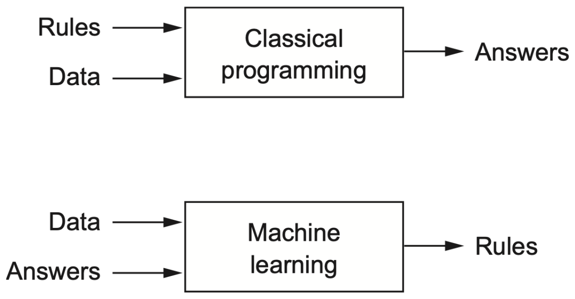


Figure 3.2: Machine Learning: a new programming paradigm. Courtesy of [25].

The success of a machine learning technique is, therefore, the capability to predict the best model that better maps the input-output relation. Given a set of models $\mathcal{M}$, it is therefore necessary to define an objective way to state which of them performs and generalizes better the model. Considering the same train set of data, it is possible to fit on them several models. Looking at the three of them in red curves in the Fig. 3.3, it is possible to notice how they differently behave:

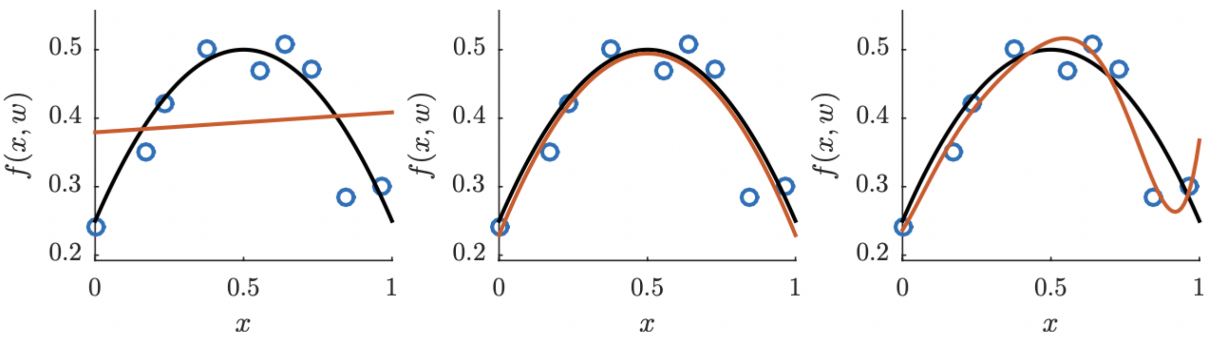


Figure 3.3: A small dataset of nine observations generated from the true curve shown with the black line. The three red lines defines three different regression models $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$ fitted to the dataset. Courtesy of [27].

It is easy to see how well the $\mathcal{M}_2$ fits the data, surely better than $\mathcal{M}_1$, but worse than $\mathcal{M}_3$, the most complicated one. However, remembering the task of creating a model capable to better predict the input-output relation, it is fundamental to observe that $\mathcal{M}_3$ will not generalize well to new data. This, is commonly known in ML field as *overfitting*. Considering a new test set of data, testing the models on these new values will provide a better estimation of how well the models *generalize* to new data. A *test error*[1] can be defined as:

$$E_{M_s}^{test} = \frac{1}{N^{test}} \sum (y_i - f_{M_s}(x_i, w))^2 \tag{3.1.0.1}$$

where f_{M_s} is the model fitted to the *training data*. The *test error* will provide a better way to estimate the generalization of the models. However, to generalize it with respect to any new possible *test set*, the *cross-validation* method could be applied. The cross-validation method takes the training set and divide it into new *train* $\boldsymbol{\mathcal{D}}^{train}$ and *test* $\boldsymbol{\mathcal{D}}^{test}$ set, and uses these two to select the appropriate model.
Indeed, considering a supervised machine learning problem with a given data set $\boldsymbol{\mathcal{D}}$, and different models $\mathcal{M}_i$ to be compared, it is possible to collect for each of them a *loss function* L that quantifies the prediction error. In general, the loss function can be different according to the purpose of the task. Then, the different models will be evaluated on different partitions of train and test sets, and the errors will be used to compute the *generalization error*, the fairest estimate of how well the model performs:

$$E_{\mathcal{M}}^{gen} = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})}[L(\boldsymbol{y}, \boldsymbol{f}_{\mathcal{M}}(\boldsymbol{x}))] \tag{3.1.0.2}$$

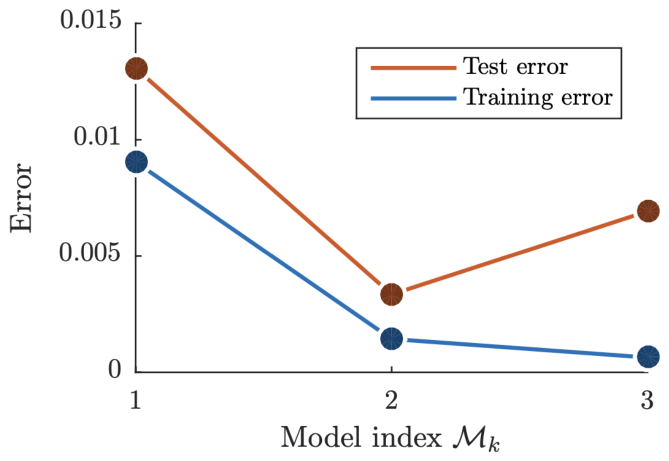


Figure 3.4: The test error for each of the three models $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$ computed on the test data set. The test error correctly singles out the $\mathcal{M}_2$ as the best model. Courtesy of [27].

When Machine Learning fails

The purpose of machine learning is therefore to find the best model that better generalizes a certain problem, and this is found by the generalization error. A model may have a high generalization error for several reasons. It may be misapplied or too weak, that is *inflexible* to learn a rich representation to solve a problem, or the model may focus on the wrong thing. Very flexible models, with little data, will learn any number of representations and then will fail: they will *overfit*. Increasing the number of data, the more flexible model

[1]The choice of the error can vary, generally depending on the type of the task or also type of data.

will be capable to learn a more sophisticated and accurate representation of the problem [25].

The choice of the models to be applied depends on numbers of boundary conditions such as the task, the type of data to be processed, the execution time needed, or the outputs. However, the methods can be divided in different categories, helping in the final choice of family of methods. It is possible to distinguish different approaches of how tasks are accomplished: *supervised learning*, *unsupervised learning* and *reinforcement learning.*

3.1.1 Supervised learning

Among the possible tasks that machine learning aims to solve, it is possible to distinguish three different learning approaches:

- in *supervised learning* the task is to learn a function that maps an input to an output based on known input-output pairs, commonly named *training data*;
- in *unsupervised learning*, given only input data $\mathbf{X}$, the objective is to infer structure in $\mathbf{X}$ such as a clustering, outlier detection, density estimation, and association mining;
- *reinforcement learning* is a learning approach used for making a sequence of decisions. The computer faces a game-like situation, employing trial and error to come up with the solution on the basis of rewards and penalties received for the action it performs.

A training set of observations $\boldsymbol{x}$ and targets y are given and the task is to come up with a model capable to map an input to an output with a function:

$$y = f(\boldsymbol{x}, \boldsymbol{w}) + \epsilon \tag{3.1.1.1}$$

where $\boldsymbol{w}$ is a vector of tunable parameters and ϵ a noise term. Therefore, learning consists in defining values of parameters $\boldsymbol{w}$ that better estimate the model on the basis of the training data. If y is continuous, the model is a *regression model.* On the other hand, if the output is a discrete one, the model will be a *classification model.*

A description of some of the *Machine Learning* techniques will be presented in the next sections, showing the insights behind their success in predicting rules. A dedicated section for *Neural Networks* will be presented later (section 3.2) to explore more in detail how this method works. For the purpose of this work, some of the main supervised learning techniques will be introduced.

Linear regression

Between the mathematical techniques, the *linear model* simply maps an input $\boldsymbol{x}$ to an output y through a *linear function*:

$$f(x, w) = w_0 + w_1 x_1 + ... + w_m x_m \tag{3.1.1.2}$$

The model is therefore a linear function of input $\boldsymbol{x}$. The purpose of learning is estimating the coefficients w_i, given the (x_i, y_i) pairs. One of the most common approaches involves the minimization of *least square*. Considering the output y and its prediction $\hat{y}$ it is possible to compute the residual $e = y$ - $\hat{y}$. The *residual sum of squares* (RSS) is:

$$RSS = e_1^2 + e_2^2 + ... + e_n^2 \tag{3.1.1.3}$$

The least squares approach chooses the learned coefficients, considering the sample means, as minimizers, that are the *least squares coefficients* for a simple linear regression.

Logistic Regression

Even though regression and classification appear very different, the linear regression can be easily extended to classification by making use of probabilities. In case of *binary* qualitative response, the task is to classify an output y as a negative class 0, or positive 1. For this purpose a general probability problem may be defined:

$$p(y|\boldsymbol{x}, \boldsymbol{w}) \tag{3.1.1.4}$$

Being y binary, the leading idea is to model its density as Bernoulli variable. Since the output of a linear model is generally a continuous number, and the θ of Bernoulli distribution belongs to $[0, 1]$, the Bernoulli distribution can be re-parameterized using a *sigmoid* function and written as:

$$p(b|z) = Bernoulli(b|\theta = \sigma(z)) = \sigma(x)^b(1 - \sigma(z)^{1-b}, \sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.1.1.5}$$

with $\sigma(z)$ the *logistic sigmoid* Fig. 3.5.

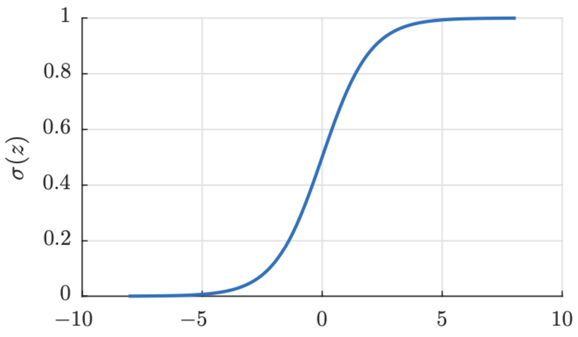


Figure 3.5: The logistic sigmoid fucnction $\sigma(z) = (1 + e^{-z})^{-1}$. Courtesy of [27]

Therefore, the probability density of a given observation y_i is:

$$p(y_i|x_i) \tag{3.1.1.6}$$

The problem turns out to be solved in the *maximum likelihood framework*. In the problem formulation stated in Eq. (3.1.1.4), $\boldsymbol{w}$ are the parameters in the model and they can be learnt by letting them equal to $\boldsymbol{w*}$ found in this case by:

$$\boldsymbol{w}* = argmin\{E(\boldsymbol{w})\} \tag{3.1.1.7}$$

However, it is not possible to solve such a formulation analytically for $\boldsymbol{w}*$: a viable approach, called *gradient descent*, can be applied and it will be presented later for *Neural Network*.

K-nearest neighbour (KNN)

Different approaches try to estimate the conditional distribution of Y given X, and then classify a given observation to the class with the highest estimated probability. Given a positive integer K and a test observation x_0, a KNN classifiers identifies in the training data the K closest points to x_0, represented by N_0. The conditional probability for class j is then estimated as the fraction of points in N_0 whose response values equal j:

$$p(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in N_0} I(y_i = j) \tag{3.1.1.8}$$

The Bayes rule (Eq. (3.1.1.9) is then applied and the test observation x_0 is then classified to the class with largest probability.

$$p(y|x) = \frac{p(x|y)p(y)}{\sum_{y'=1}^{c} p(x|y')p(y')} \tag{3.1.1.9}$$

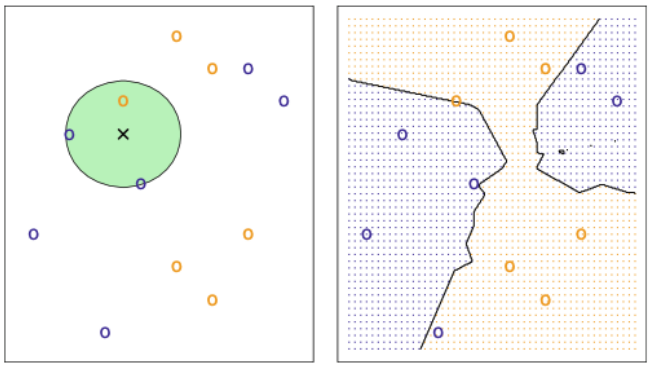


Figure 3.6: The KNN approach using $K = 3$. On the left a test observation to classify is shown in black cross. The three closest points to the test observation are identified, and the test is classified according to the most occurring class in blue, with a rate of 2/3. On the right, the decision boundary is shown in black. The blue grid region tells where a test observation will be classified as blue class. The same is applied for the yellow class. Courtesy of [28]

3.2 Deep Learning and Neural Networks

Deep learning is a specific subfield of *machine learning* that puts emphasis on learning successive layers of increasingly meaningful representations. The concept of *deep* comes from the idea of successive layers of representations where, their number, indeed, defines the *depth* of the model. The layered representations are learned via models called *neural networks*, a name that references to neurobiology and the structure of neurons in the brain. The need of adopting *Neural Networks* increased in the last decade thanks to the increase of amount of data available and computational power.

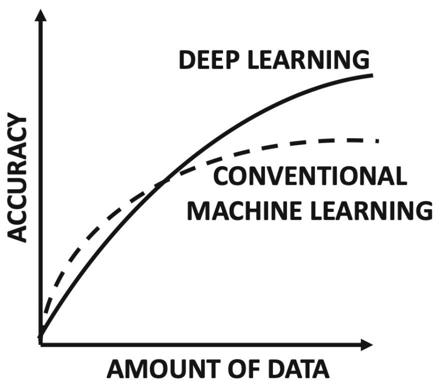


Figure 3.7: Qualitative comparison of the accuracy of a typical machine learning algorithm with that of a large neural network. Deep learning methods become more attractive when sufficient data/computational power is available. Courtesy of [29]

In a Neural Network architecture it is possible to define a set of information processing units, called *neurons*, and each of them is connected to other neurons by weighted connections. They are collected into *layers* with connections from one layer feeding into the next.

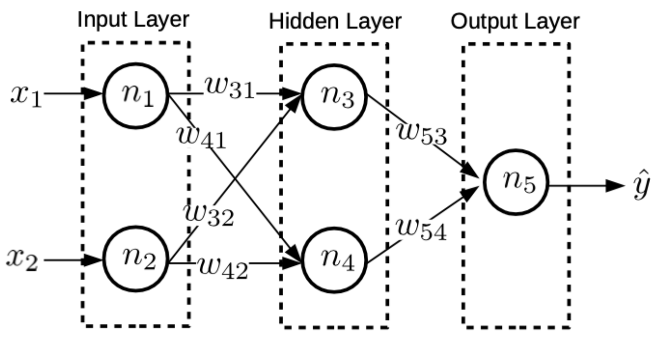


Figure 3.8: A simple neural network with 6 weights and one hidden layer with 2 neurons. Courtesy of [27]

The process of learning in a *neural network*, consists of defining the set of weights that better estimate the model mapping an input to an output. Two are the main phases of learning:

- *Forward propagation*: the input data for a training sample are fed into the network. A cascade of computations across the layers is carried out, using the actual values of the weights. The weighted sum of the value of each neuron is fed into the next layers and then *activated.* The final output is computed and compared to the real target with a *loss function.* The loss function and the weights will be stored for the next phase.
- *Backward propagation*: the objective is to learn the gradient of the loss function with respect to the different weights using the chain rule of differential calculus, allowing to determine the influence of the weights on the loss. The derivative of the loss will be

then used to update the weights to properly minimize the loss score: in this phase the learning happens.

Given the *loss function f*, typically a differentiable one, it is theoretically possible to find its minimum. The objective is therefore to analytically find the combination of *weights* that yields the smallest possible loss score. This is done by solving the $\nabla f = 0$ for w. To deal with the large amount of *weights* a neural network may present, an alternative is obtained by updating the weights in the opposite direction from the gradient, reducing the loss step by step every time applying therefore the *gradient descent* method.

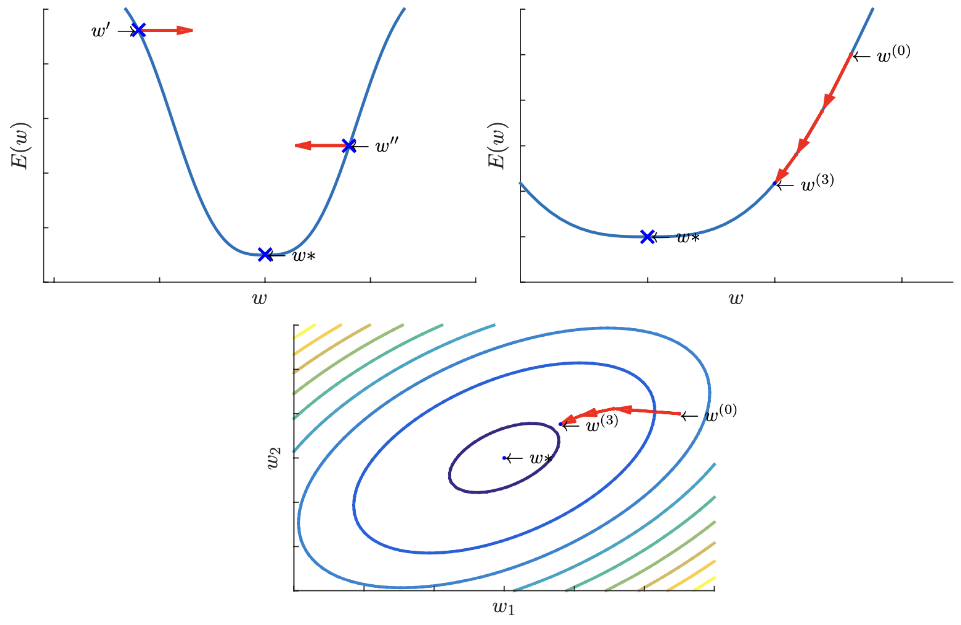


Figure 3.9: On the top-left: error function *E(w)* for 1D example. Weights at *w'* should move right and *w"* to left in order to approach the minimum. On the top-right: the gradient descent algorithm is applied for three steps starting at $w^{(0)}$. On the bottom: value of the error function in a 2D dimensional example as a contour plot along with three steps of the gradient descent algorithm. The step size slows down when approaching the minimum of the function. Courtesy of [27].

The weights may be assigned in different ways, randomly, equal to zero or in other ways like following the Xavier initialization. Therefore, the output will be initially far from what it should really be, with a high loss value. However, every example the network processes, allows the weights to be adjusted, achieving a better loss score. The *training loop*, repeated a sufficient number of times over thousands of examples, will lead to the right weights values for loss decrease.

3.2.1 Activation functions

Activation functions are a central part of the design of a neural network. The choice of the right activation function defines how well a certain network will perform and learn. For what concerns the output layer, the choice of the activation function will define also the type of prediction the model can make. They define basically how the weighted sum of the input is transformed into an output from nodes in a layer, allowing the construction of a model capable to deal with non-linear input-output relations. Different may be the functions:

- **ReLU** Rectified linear activation function is probably the most common function used for hidden layers. It is less susceptible to vanishing gradients that prevent models from being trained, but may suffer however of saturated units problem. It is presented in the form of $f(x) = max\{0, x\}$;

- **Logistic function** It is the same function used in logistic classification algorithm. It is usually fast in classifying unknown records being prone, however, to construct on linear boundaries. In addition it would face saturated gradient since it ranges only between 0 and 1;

- **tanh** The hyperbolic tangent activation function has a similar shape to LogReg's one. It takes any input and outputs it into [-1,1], being centered on 0. However, like *sigmoid*, its activation may saturate;

- **Leaky-ReLU** Leaky-ReLU tries to fix "dying ReLU" for saturated units. For $x < 0$ the function will have a small slope. It is in the form $f(x) = max\{\alpha x, x\}$, with α tipically 0.01;

- **maxout** The maxout neuron, introduced by *Goodfellow et al.* [30] computes the function $f(x) = max\{w_1^T x + b_1, w_2^T x + b_2\}$, getting the advantages of a ReLU (linear regime of operation) and not the disadvantages of dying ReLU. It, however, leads to a high total number of parameters.

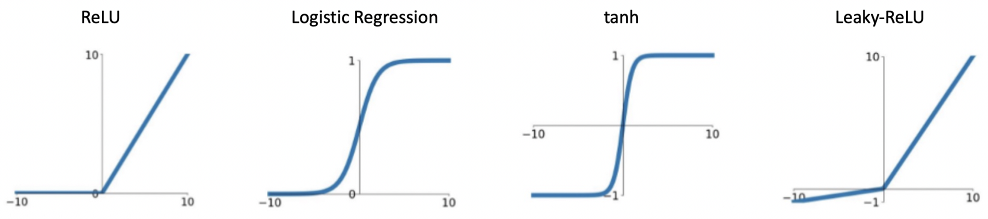


Figure 3.10: Some of the most used activation functions.

3.3 Neural Networks for Image Processing

The strong success *neural networks* had in the years, dealing with higher amount of data, allowed researchers to develop different variations and architectures to deal with certain types of data. In particular, *Convolutional Neural Networks* (CNN) had important results in the field of image recognition, ranging from object and features detection and localization, up to text processing. A CNN is capable to successfully capture the spatial and temporal dependencies in an image through the application of particular filters. Its role is therefore to reduce the images into an easier form to process without loosing the features that are critical for achieving a good prediction, learning local patterns. The main idea of *convolutional networks*, therefore, relies on convolution step involving application of filters.

The motivation for the convolutional networks was introduced by *Hubel and Wiesel*'s understanding of the workings of the cat's visual cortex [31]. The first basic architecure

based on this biological inspiration was the *neocognitron*, generalized later to the *LeNet-5* architecture [32], introduced by *Yan LeCun* from Bell Labs. The idea was applied to the problem of handwritten digits classification. Considering its success and accuracy in the task, the model was then used by USA Postal Service in the 90s to automate the reading of ZIP codes on mail envelopes. However, the CNNs did not spread at the time: they needed lots of data and computational resources to work efficiently for large images.
In the 2000s, the increasing spread of personal computers and internet as data source, and the availability of more powerful computation tools like parallel computing offered by GPUs, allowed the development of more sophisticated and accurate networks. In 2012, *Alex Krizhevsky et al.* introduced a deep CNN, called *AlexNet* [33], that successfully competed in the *ImageNet Large Scale Visual Recognition Challenge*, making use of a GPU to allow the implementation of a 8 layer model. It showed that time had come to revisit deep learning.

3.3.1 Convolution layer

The *convolution layer* is the core block of the CNN. It performs a dot product between two matrices, where one matrix is composed by the learnable set of parameters, also known as kernel, typically smaller than the input, and the other is the restricted portion of the receptive field. During the forward pass, the kernel slides across the height and the width of the image. This produces a 2D representation of the image that gives the response of the kernel at each spatial position of the image, called *activation map.* The idea on which a CNN relies on is to use the kernel properly as a *filter* capable to map particular features in the image, starting from the simplest ones like vertical or horizontal edges reaching more complex structure moving deeper in the network.

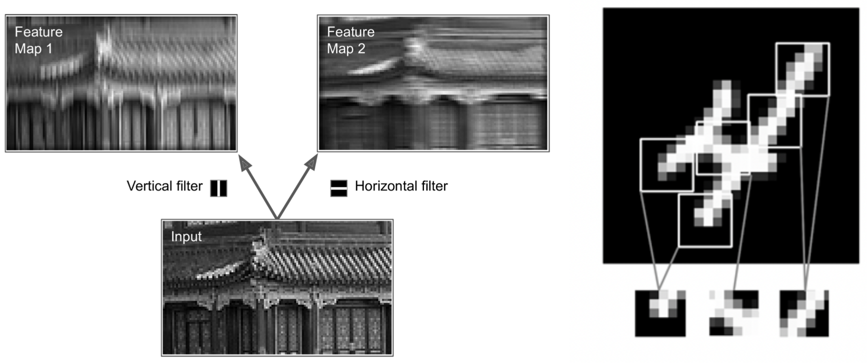


Figure 3.11: On the left: applying two different filters to get two different feature maps. Courtesy of [34]. On the right: images can be broken in local patterns such as edges, textures, and so on. Courtesy of [25].

Operating such a way, allows CNN to get interesting properties. Firstly, the patterns they learn are *translation invariant.* A learnt pattern in the lower corner of a picture, will be recognised anywhere. A densely connected layer would have to learn anew if it appeared in a new location. This allows the CNN to be efficient in image processing, requiring fewer training examples. Secondly, they learn *hierarchies of patterns.* As previously said, the first layer will learn small local patterns like edges, the second will learn larger patterns made of the features of the first layers and so on. CNN learn therefore increasingly complex and abstract visual concepts.

In the forward propagation, the filter, typically of size 5×5 or 3×3, with the same depth of the input, slides onto the input image. The convolution operation reduces the size of the input matrix because of the size F of the filter and the stride S, the spatial step of the sliding kernel on the input. To not to lose information, the *zero-padding* strategy is involved: a certain number of rows and columns of zeros is added to the input matrix in order to not to lose the information on the image's edges due to the application of the kernel.
A CNN therefore requires an accurate choice of the *hyperparameters* defined as the kernel size F, the number of filters K, the stride S and the padding P.

3.3.2 Pooling layer

Being the convolution step a linear process, the convoluted output is usually fed into an activation function (already presented in section 3.2.1) to introduce non-linearity to the *activation map*. Then, a *pooling layer* replaces the output of the network at certain locations by deriving a summary statistics of the nearby outputs, helping reducing the spatial size of the representation, decreasing the number of computations and weights. Among the different *poolings* the most common ones are the *max pool* (Fig. 3.12) and the *average pool*.

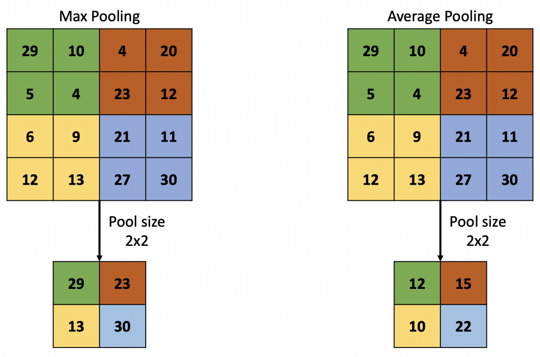


Figure 3.12: Max pooling operation: it returns the maximum value from the portion of the image covered by the kernel. It performs also as a *noise suppressant*. Average pooling operation: it returns the average of all the values from the portion of the image covered by the kernel.

3.3.3 Fully connected layer

Neurons in this layer have a full connectivity with all the neurons in the preceding and succeeding layer like in a FCNN. It helps in mapping the representation between the input and the output.

After data are processed by the FC, they are ready to be read by other functions to accomplish the task of the model (e.g. a softmax for multi-classification). The procedure of learning is similar to the one explained section 3.2: a loss function will be used as metric for learning and its gradient used for updating *weights* in the *backpropagation*. The Fig. 3.13[2]

[2]https://towardsdatascience.com/image-classification-in-10-minutes-with-mnist-dataset-54c35b77a38d . Last accessed: 24-11-2021.

shows a typical structure of a CNN for multiclassification purpose:

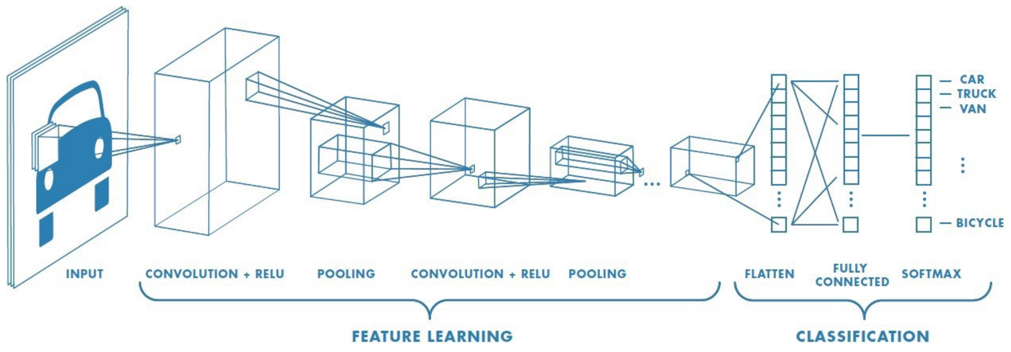


Figure 3.13: A typical structure of a Convolutional Neural Network. The image is input into the network. The pixel matrix is convolved by a a kernel that provides the activation map. Non-linearity are the introduced in the network by a ReLU function. The output matrix is fed into a pooling operation. The scheme may be repeated such that the next convolution layer will learn more high level features. The output is then flattened and a fully connected layer is introduced. A *softmax* operation is applied to perform a classification task.

3.4 Residual Networks

The success *AlexNet* [33] obtained in the classification task was accredited to its additional layers, made by 5 convolutional layers and 3 fully-connected ones. This led a period of development of deeper networks following the success of *Krizhevsky et al.* [33]. The intuition behind it was that the layers progressively learned more complex features: the depth of a network seemed to be of crucial importance for the performance of a model. In 2016, *He et al.* from Microsoft [35] tried to answer a question: *Is learning better networks as easy as stacking more layers?* They plotted a comparison of *training and test errors* of a 20 and 56-layers models (Fig. 3.14):

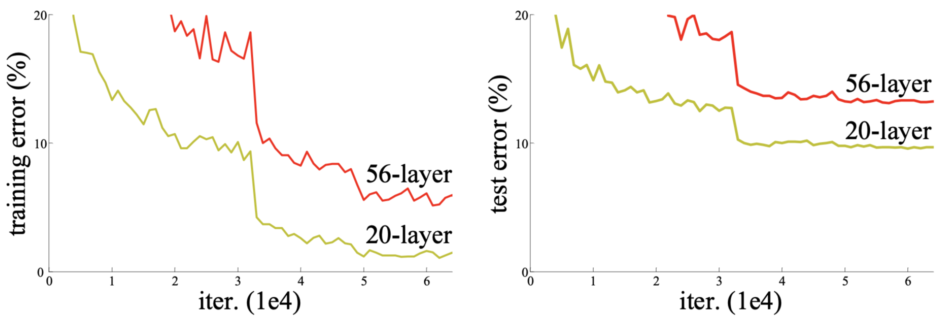


Figure 3.14: Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer plain networks. The deeper network has higher training error, and thus test error. Courtesy of [35].

The plots showed that deeper networks presented higher train and test errors. The failure of the deeper network was neither due to vanishing/exploding gradients, since this would have been easily addressed via normalized initialization and intermediate normalization layers, nor to *overfitting*, easily solved by L2-norms or dropouts: adding more layers simply led to higher *training errors*.

To address the issue observed into Fig. 3.14, *He et al.* presented the idea of *Residual Network (ResNet)*:

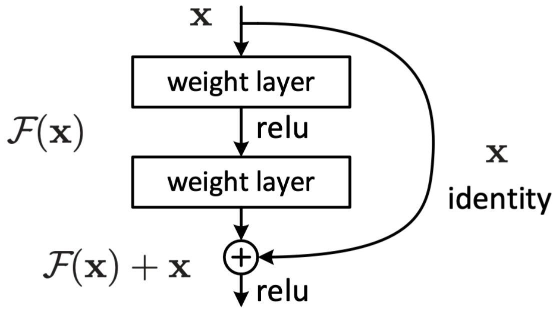


Figure 3.15: Residual learning: a building block. Courtesy of [35].

Instead of having each few stacked layers directly fitting an underlying mapping, they fit a *residual mapping*. Considering the desired underlying mapping, namely $\mathcal{H}(x)$, the stacked non-linear layers fit another mapping of $\mathcal{F}(x) := \mathcal{H}(x) - x$. The original mapping is recast into $\mathcal{F}(x) + x$. What *He et al.* [35] obtained was that it was easier to optimize the residual mapping instead of the unreferenced mapping. The formulation of $\mathcal{F}(x) + x$ is realized by a feedforward neural network with a *shortcut connection* that skips one ore more layers. This could be simply an *identity mapping*. With the idea of learning not the real mapping but its residual, introducing this novel model, *He et al.* obtained successful results:

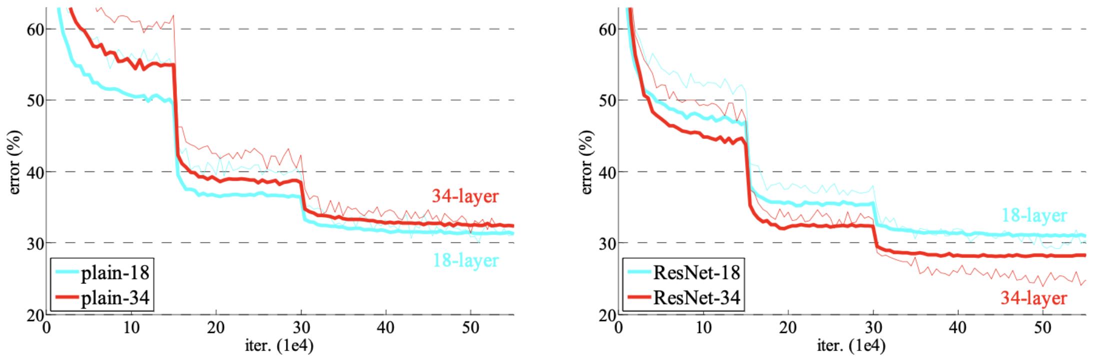


Figure 3.16: Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers without skip connections. Right: ResNets of 18 and 34 layers. Courtesy of [35].

Using skip connections, it was possible to easily train even deeper networks, obtaining better training and test error with an increased number of layers (e.g. *He et al.* obtained excellent results for 152-layers network), allowing the model to learn more sophisticated features and not loosing performances.
The high performances of such a model allows ResNets to be actually applied for number of tasks like semantic segmentations, Generative Adversial Networks and more.

3.4.1 Convolutional Neural Network for Centroiding: MarsNet

The huge impact Artificial Intelligence had on the modern technologies spread throughout different fields, maturing into well proved industrial products. The techniques behind *machine learning* engaged the attention of scientists in the last decades.
In particular, also space sector has been trying to align to the development of such technologies, incorporating in its products the concepts related to AI. Today, applications of interest range from preliminary spacecraft design to guidance and control algorithms over navigation and the prediction of the dynamics of perturbed motion, and even classification of astronomical objects.

Differently from the solutions and the applications that have been presented in the introductory chapter of this work, a different method, based on Convolutional Neural Networks model, has been presented by *Thibaud F. Theil* in the work *Optical Navigation using Near Celestial Bodies for Spacecraft Autonomy* [36]. By making use of a CNN model modified as a ResNet, the work focuses on navigation in proximity to a known celestial body. In particular, the author chose a spacecraft orbiting around Mars as working scenario. The objective was to investigate the capabilities of a CNN model to perform Center and Apparent Diameter measurements. Therefore, through a regression method, the model outputs the two coordinates of the center and the radius of Mars in the pictures by reading the input image.
The higher level architecture represents a series of down convolutions, where not every pixel is sampled in order to decrease the dimensional size of the image while increasing the perceptive field of each feature pixel. Then, the convolutions are input to the ResNet Blocks, which are then followed by a linear output layer that provides the prediction. In particular:

- the first layer applied is a 2D-Convolutional layer which takes in 3 channels, output 16, with a 3×3 kernel, 1×1 stride and padding;
- A set of five ResNet sequence blocks are then defined each followed by a 2D, down sampling convolution;
- the ouptut is then flattened, linearized and activate by a Leaky-ReLU function.

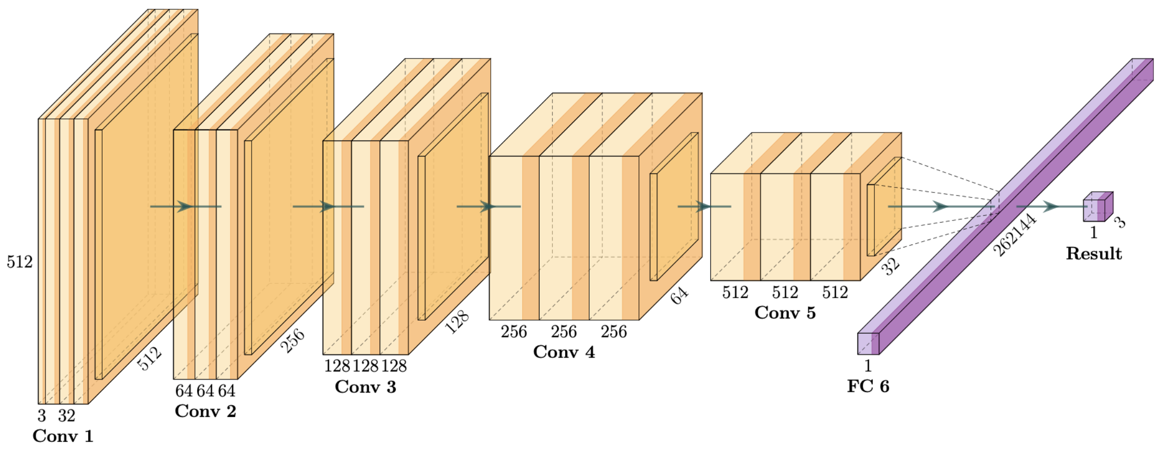


Figure 3.17: The MarsNet architecture applied in the work of Thibaud F. Theil. Courtesy of [36].

The author preferred an adaptive optimization algorithm such as the Adam over the Stochastic Gradient Descent. Even though the Adam tends to perform well in the initial part of training, it is usually outperformed by SGD at later stages. Nevertheless, the Adam was chosen for the training to achieve fast training along with high performances, using, in addition a Leaky-ReLU as activation function.
The author, used the Huber loss, being less sensitive to outlier predictions from the network, allowing, in addition, to use even larger learning rates avoiding exploding gradients
For each dimension size, three ResNet block were applied, each repeating twice the following operations:

- Apply a 2D-Convolution to the input;
- Activates the convolved input with a Leaky-ReLU;
- Batch-normalization of the activated layer;
- A 0.5 dropout, zeroing each neuron with probability one-half;

The model was trained for 40 epochs with a batch size of 14 using a learning scheduler that reduced the learning rate after 5 epochs without improvements to perform a regression task for the center and radius estimation.

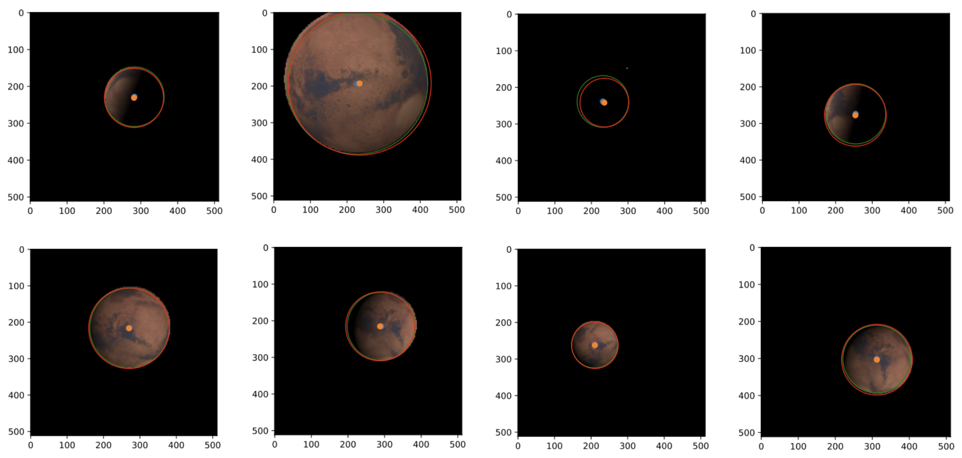


Figure 3.18: MarsNet CNN model evaluated on Mars images. Courtesy of [36].

Chapter 4

Center and Apparent Diameter techniques

The previous chapters presented an overview of the main techniques that have been applied in the past but also still under development. The objective of the work is to understand how and if a *Neural Network* architecture can outperform the classical approaches of *Image Processing* for *Optical Navigation*.
Among the solutions, the Center and Apparent Diameter (CAD) techniques have been chosen to be compared. More in detail, the comparison will be carried out between:

- the CAD technique presented in section 2.3.2, involving the *improved ellipse fitting with direct least-squares estimation* to retrieve the average radius and the center of the ellipse of the Moon's body in the image, relying on a method robust in numerical instability;
- the CAD technique performed by a convolutional neural network based on the idea of *MarsNet*, to carry out a regression task for radius and center prediction.

To be coherent with the LUMIO mission, it has been necessary to develop an image dataset that could be as realistic as possible with respect to the mission environment.

4.1 Image Dataset Generation

The development of the images of the Moon has been carried out by making use of *Blender*[1], an open-source 3D rendering software supporting *Python* scripting, helpful for automating processes. Moreover, it provides the option to work on different internal rendering engines like *Eevee, Workbench* and *Cycles* or external ones, achieving different quality solutions. The chance of using *Python* in the software, allows to automate image capturing processes, in different positions of the camera with respect to the Moon, and with different illumination conditions accordingly to the official orbit coordinates.

[1]https://www.blender.org . Last accessed: 11-11-2021

4.1.1 Dataframe setting

To take advantage of the *Python* scripting in *Blender*, it has been necessary to create the *dataframe* documentation containing the coordinates of LUMIO spacecraft. Firstly, a *reference frame* has been chosen to describe the position of the spacecraft with respect to the Moon. A suitable solution was a *Body Fixed Reference Frame* aligned with Moon's *Principal Axis* which allowed to ease the tracking process of the camera and the Sun in *Blender*. The official coordinates values for the mission were provided in EME2000 for the Sun, the Moon and LUMIO. Therefore, the rotation matrices have been computed for each epoch, expressed in seconds, available in the datasheets, to perform a transformation from EME2000 to PA. SPICE[2] has been used in the MatLab environment to perform the computations and to obtain the matrices for the transformation to MOON_PA_DE440.
By triangulation, firstly, the position of the Sun and LUMIO have been retrieved and expressed with respect to the Moon position. Secondly, the obtained vectors have been transformed in the MOON_PA reference frame to retrieve the correct camera view and illumination condition of the Moon.
All the coordinates information have been stored in a new dataframe in .csv format, providing the needed data to be passed to *Blender* to automate the rendering. For this purpose, the *csv* and *pandas* packages have been used to create and modify the datasheets in *Jupyter Notebooks*[3].

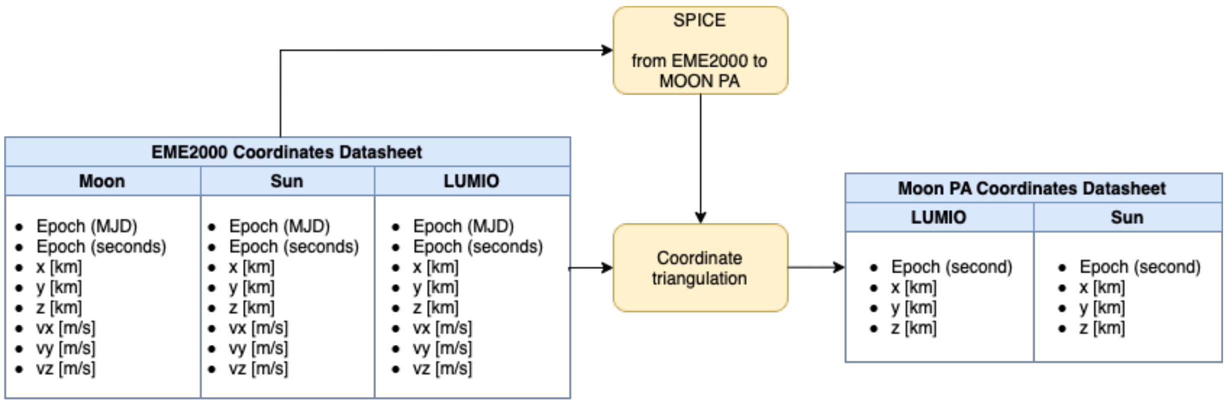


Figure 4.1: Datasheet generation.

[2]https://naif.jpl.nasa.gov/naif/ . Last accessed: 11-11-2021
[3]https://jupyter.org . Last accessed: 23-11-2021

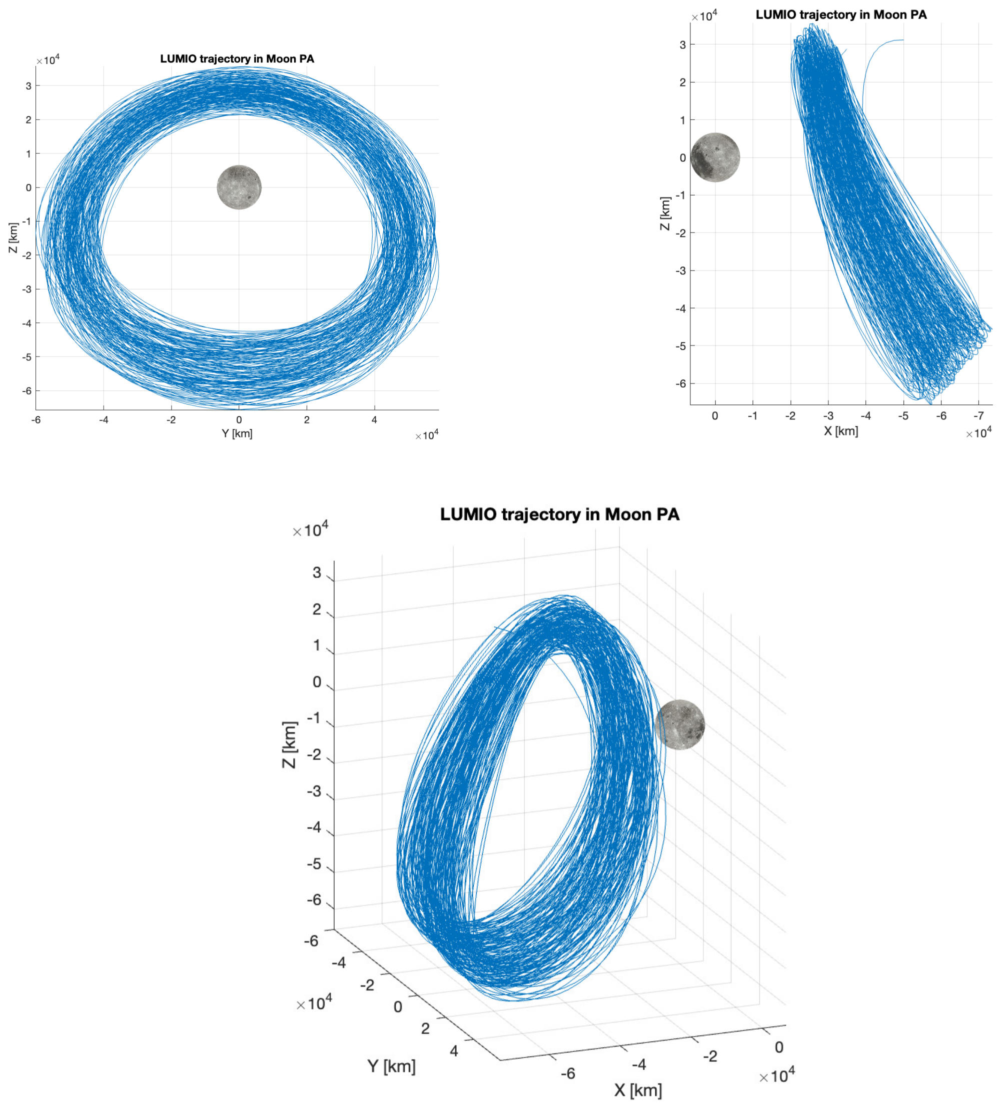


Figure 4.2: LUMIO orbit in Moon Principal Axis Reference Frame.

4.1.2 Camera and Environment Setting

To simulate a scenario as similar as possible to the one of the mission, the camera and the environment in Blender have been accordingly set up. The volume of the Moon has been created as *UV Sphere* on Blender. To achieve a realistic model, the sphere has been modified in the dimensions to obtain an ellipsoid rather than a simple sphere, on the basis of the Moon's sizes[4]. This allowed to perform the image processing techniques on a pseudo-real mission scenario.

Polar diameter [km]	**Equatorial diameter [km]**
3472.0	3476.2

Table 4.1: Moon physical parameters.

[4]https://nssdc.gsfc.nasa.gov/planetary/factsheet/moonfact.html . Last Accessed: 11-11-2021

The colormap and the displacement map have been retrieved and applied from the CGI MOON KIT of *NASA Scientific Visualization Studio*[5] and oriented accordingly to the MOON_PA_DE440. A good trade-off between resolution quality and rendering speed has been achieved with an $8K$ colormap and unsigned 16-bit TIFF displacement map. A number of samples to render for each pixel has been selected to be equal to 32 allowing a fast and accurate rendering.
The camera parameters have been set up accordingly to the official values provided by [18], allowing to render 1024×1024 gray-scale squared images centered on the Moon.

Resolution	**Field of View**	**Focal length**	**Pixel size**
1024×1024 pixels	6.0 deg	127 mm	13 μm

Table 4.2: LUMIO-Cam properties.

Illumination *Blender* allows to choose between four possible options of light settings: the *point* light that is an omni-directional point of light, a point radiating the same amount of light in all directions; *spot* lights that emit a cone-shaped beam of light from the tip of the cone, in a given direction; *area* light simulating light originating from a surface emitter; *sun light* that provides light of constant intensity emitted in a single direction from infinitely far away[2]. Considering the nature of the case study, the *sun light* object has been chosen to illuminate the surface of the Moon. In this case, its position does not count but only the direction of the emitting rays.

Render engine The choice of the render engine has been done between *Cycles* and *Eevee*. *Cycles* is *Blender*'s physical-based path tracer for production rendering, that creates an image by tracing the paths of rays through the scene, providing highly detailed results coming at cost of the time-processing. On the other hand, *Eevee* is a real-time render engine focused on speed and interactivity, achieving the goal of rendering PBR materials. Instead of computing each ray of light, it uses rasterization, that estimates the way light interacts with object through combination of algorithms[2]. It produces, however, less accurate images but rendered in almost real-time.
Despite the time consuming process, *Cycles* has been chosen for rendering, to achieve better and more accurate images, especially for detailed terminator in the Moon's picture.

4.1.3 Image dataset

Given the coordinates of LUMIO and the Sun in the Moon PA, the images have been accordingly rendered. As objects of comparison between the two algorithms, a *test set* $\mathcal{D}^{test}$ has been generated on the basis of the official LUMIO coordinates. It consists of 8935 gray-scale .png 1024×1024 squared images centered on the Moon. To emulate the scenario of the camera motion on the orbit, both the Sun and the camera itself have been set to track the Moon object in Blender accordingly to the coordinates in principal axis.

[5]https://svs.gsfc.nasa.gov/cgi-bin/details.cgi?aid=4720 . Last accessed: 11-11-2021

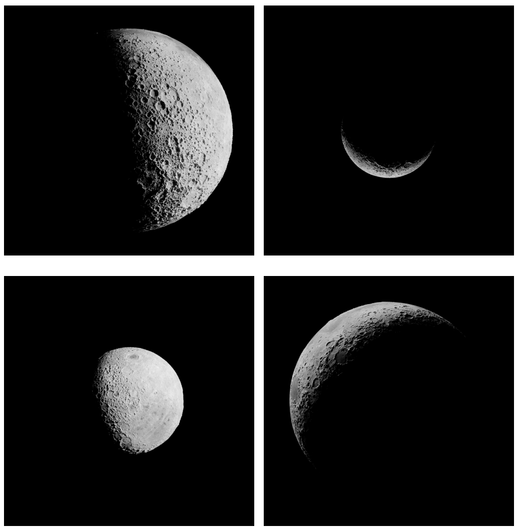


Figure 4.3: 4 samples from the Moon Data Set.

To provide the images for training for the *neural network*, a fictitious toroid of points has been built around the *test set*. For each point of $\mathcal{D}^{test}$ a sphere of random range has been created upon which 14 other points have been randomly distributed. Therefore, the final *train set* $\mathcal{D}^{train}$ made by 125090 new images has been obtained.

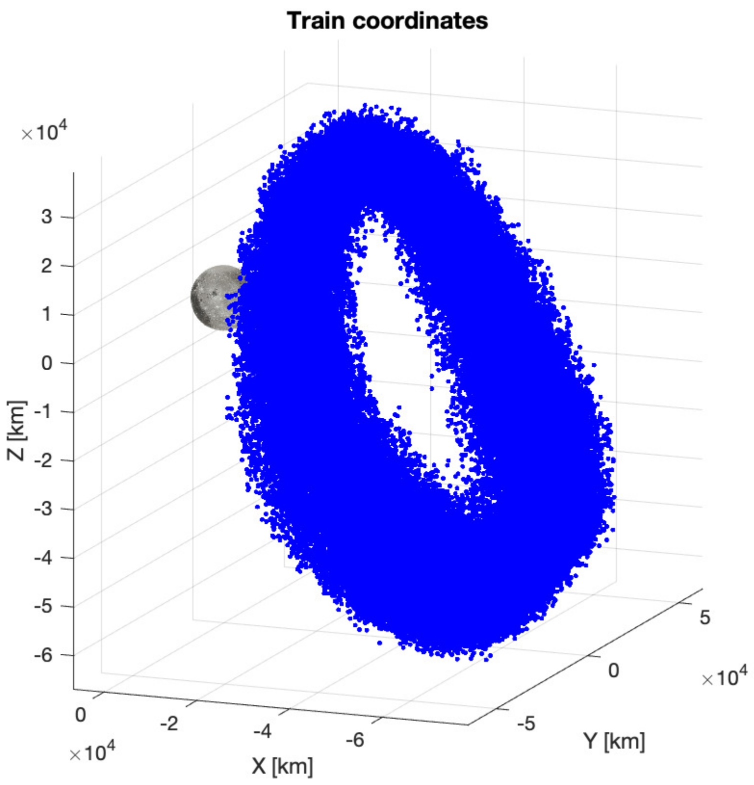


Figure 4.4: Train set coordinates points. It has been built as a fictitious toroid around the test set points. The Moon in the picture is scaled up for better visualization.

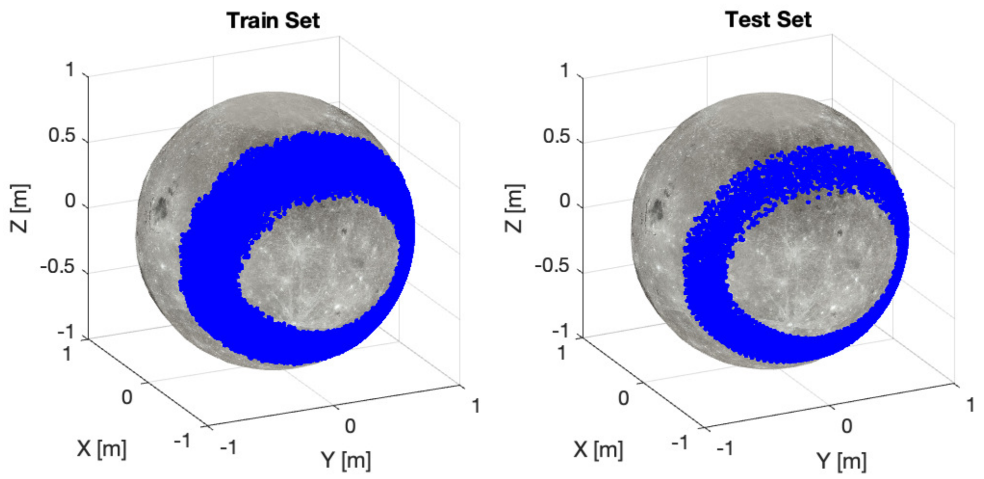


Figure 4.5: Projections of the position vectors on the Moon's surface. The coordinates have been normalized for this visualization. The farside of the satellite is the main viewed surface.

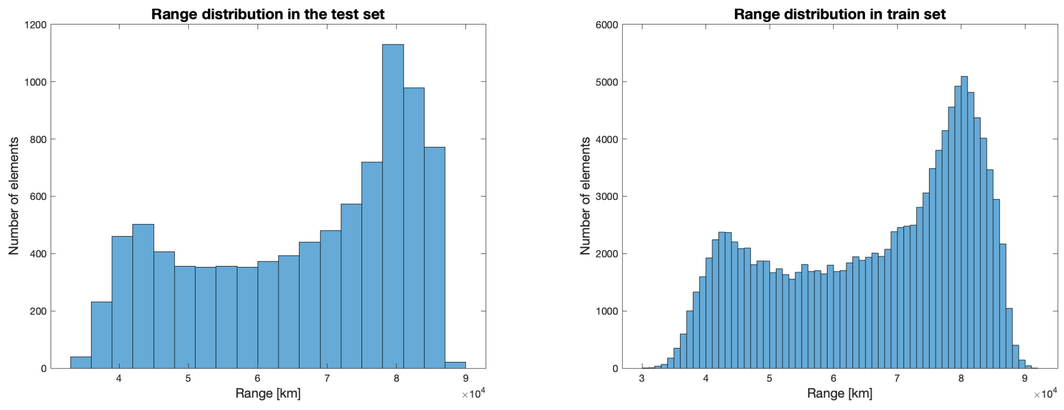


Figure 4.6: Range distributions in the test and train set.

The rendering has been performed on a MacBook Pro 2020 M1 with 8 *GB* of RAM, with *Blender* 2.93.0 run natively on Apple Silicon. The average times of rendering and the total time required are reported in the Tab. 4.3.

	Test set	**Train set**
Average Rendering Time per image [s]	5.52	5.49
Total Rendering Time [h]	13.7	190
Data Size [GB]	3.08	43.03

Table 4.3: Rendering times and data size for test set and train set.

4.2 Standard OpNav CAD

When the Moon appears to be sufficiently larger than a blob of pixels, *Centroid and Apparent Diameter* techniques can provide a good estimation of the line-of-sight and the distance of the satellite to the Moon. The process of CAD estimation has been performed on the basis of the *improved ellipse fitting with direct least-squares estimation* presented in section 2.3.2 to achieve an accurate estimation of the center of mass and radius of the displayed Moon.

4.2.1 Moon Image Processing

To perform an ellipse fitting, it is necessary to provide to the model the points in the image plane forming an arc of the horizon. However, the distinction between the horizon and the terminator of the Moon is a critical problem that has to be dealt with care. An intuitive approach to distinguish the two arcs would be to identify the direction of the sun light. A solution may be provided by the sun sensor, which would give the direction of the incoming light of the sun. However, a different approach can be used. Considering the picture of the Moon on the image plane, if the distance of the satellite to the moon is neglected with respect to the one of LUMIO-Sun, it is possible to retrieve the direction of the sun light and the axis of symmetry of the illuminated surface directly from the picture [37]. Additionally, if the process is completely based on the image, it may help in reducing the issues of task prioritization for the on-board processing unit and copying data from other sensors in the memory pool.

Eigenvalue analysis

Given the image of the Moon, it is possible to compute the principal axes of the illuminated area. Following the steps proposed by *Mortari et al.* [37], for this purpose, the image is firstly binarized by choosing a threshold gray-tone value, chosen on the basis of the image by making use of the Otsu's method [38]. The binarized image pixel matrix allows to compute the center of mass of the illuminated area as:

$$\mathbf{r_b} = \frac{\sum_i \mathbf{I r_i}}{\sum_i \mathbf{I_i}} \qquad \mathbf{c_b} = \frac{\sum_i \mathbf{I c_i}}{\sum_i \mathbf{I_i}} \tag{4.2.1.1}$$

where $[\mathbf{r_i}, \mathbf{c_i}]$ are the coordinates of a generic pixel and $\mathbf{I_i}$ is the lumped-mass binary value with the index spanning from 1 to 1024. Accordingly, it is possible to recover the inertia tensor associated to the pixel matrix:

$$\mathbf{T} = \begin{bmatrix} \sum_i \mathbf{I_i}(\mathbf{r_i} - \mathbf{r_b})^2 & -\sum_i \mathbf{I_i}(\mathbf{r_i} - \mathbf{r_b})(\mathbf{c_i} - \mathbf{c_b}) \\ -\sum_i \mathbf{I_i}(\mathbf{r_i} - \mathbf{r_b})(\mathbf{c_i} - \mathbf{c_b}) & \sum_i \mathbf{I_i}(\mathbf{c_i} - \mathbf{c_b})^2 \end{bmatrix} \tag{4.2.1.2}$$

The largest positive eigenvalue of $\mathbf{T}$, λ_{max}, allows to compute the inclination θ_{sym} of the axis of symmetry of the illuminated area as:

$$\theta_{sym} = atan2(\hat{\boldsymbol{\omega}}_{max}(1), \hat{\boldsymbol{\omega}}_{max}(2)) \tag{4.2.1.3}$$

with $\hat{\boldsymbol{\omega}}_{max}$ the eigenvector associated to λ_{max}.
The axis of symmetry is obtained as the line passing throught the center of mass $[\mathbf{r_b}, \mathbf{c_b}]$ with slope $m = tan\ \theta_{sym}$. In addition, the ratio between the eigenvalues $(\lambda_{min}/\lambda_{max})$ provides an approximated measure of how much the target surface is illuminated: a ratio close to one indicates an almost full illuminated Moon, while a ratio close to zero indicates a barely illuminated surface.

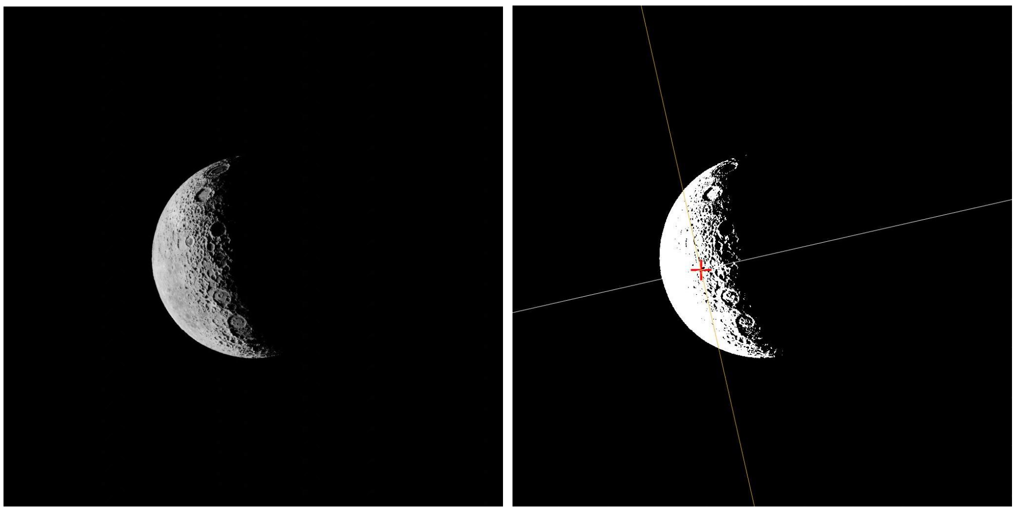


Figure 4.7: On the left: a picture of the Moon is taken by the camera. On the right: the image is binarized and the eigenanalysis step is performed and the axis of symmetry is retrieved.

Image Rotation and Horizon Detection

After the eigenanalysis is performed, the horizon and the terminator can be distinguished. Intuitively, as suggested by *Christian et al.* [2], rather than scanning the original image diagonally across discrete pixel locations, the image can be rotated and scanned horizontally row by row or vertically columns by columns in an easier and faster way. In this case the rotation of the image is performed to obtain always the lit horizon on the most up part of the picture. This has been done by bilinear interpolation through the MatLab's Image Processing Toolkit.
From the eigenanalysis, the slope of the axis of symmetry m, the angle θ_{sym} and the center of mass of the image are then used to compute the rotation angle needed to rotate the image. This is done by distinguishing different possible combinations of these values, studying the sign of θ_{sym}, the sign of m and the position of the center of mass that, together, give information about where the sun is coming from, allowing to perform the image rotation.

After the binarized and rotated image has been computed, it is firstly closed by a simple *closing operation* with a sequential operation of erosion and dilation algorithms to ease the detection processes with smooth edges in the nearby of horizon-terminator corner. Therefore, the *Canny* method is applied to distinguish the edges of the illuminated surface, identifying the horizon and the terminator. A scan of the pixel is performed column by column: every time an illuminate pixel is encountered for the first time at the column j, all the rows below at that j are set at 0 value. In this way all the points belonging to the lit horizon are saved and back rotated to the initial orientation. In addition, the closing algorithm allowed to have the certainty to delete any possible pixel below the horizon.

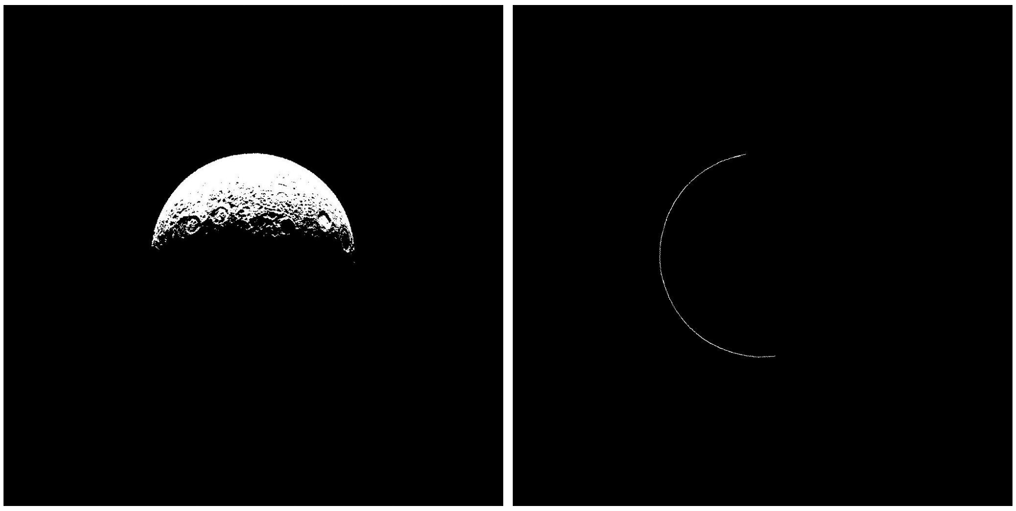


Figure 4.8: On the left: the binarized image of the Moon is rotated. On the right: the horizon points are computed and back rotated to the original image orientation.

Ellipse Fitting and Range Estimation

After the horizon has been detected, the coordinates of the illuminated pixels are used to perform the *improved ellipse fitting with least-squares estimation* (section 2.3.2). In this way the eigenvalue problem is solved for the $\mathbf{a_i}$ vectors and the coefficients of the ellipse equation are retrieved. The whole ellipse is then obtained from the implicit equation $F(x, y) = Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$.

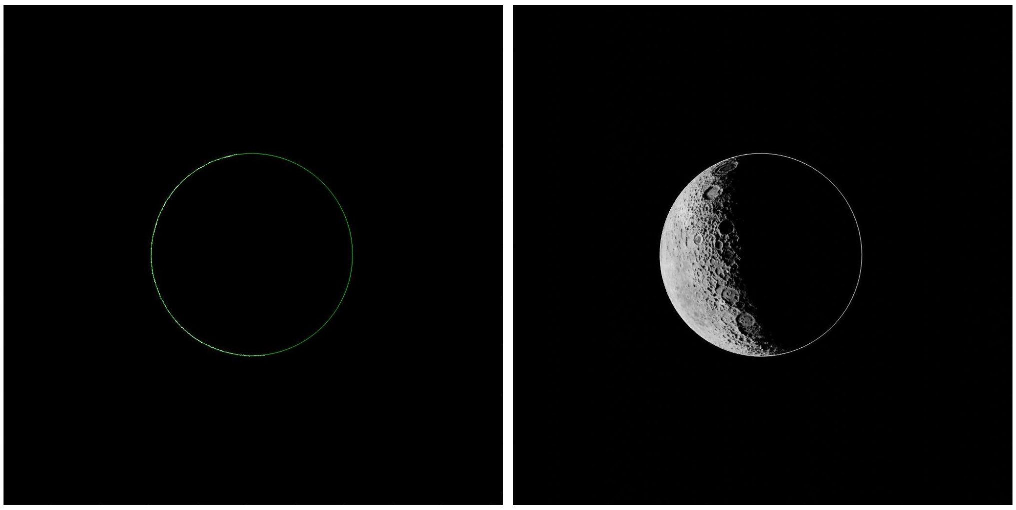


Figure 4.9: On the left: the ellipse is reconstructed on the basis of the arc. On the right: the computed ellipse is superimposed to the original image of the Moon.

After the coordinates center of the ellipse and its semi-axes are computed by application of the classical geometry formulation (Eq. (2.3.2.2), Eq. (2.3.2.3)), the radius of a circumference is retrieved from the average values of the semi-axis of the ellipse:

$$r_{avg} = \frac{a+b}{2} \tag{4.2.1.4}$$

This allows to compute the values of the radius and the center in number of pixels for the CAD algorithm.

Considering the *focal length* $f = 0.127m$ of the camera, the *pixel size* of $13\mu m$ and the average radius of the Moon of $R = 1738000\,m$, the range d of LUMIO from the Moon is estimated from the pin-hole model as:

$$d = \frac{f\,R}{r_{avg}\,px_{size}} \tag{4.2.1.5}$$

4.3 Convolutional Neural Network

The idea behind the MarsNet approach, to estimate the radius and the center of the planet, in this case the Moon, has been applied here as comparison method in the deep learning framework.
However, particular operations have been needed before the application of the network to the generated dataset. As previously said, the image data are 1024×1024 pixels sized. With such dimensions, no network would be capable to provide good performances in terms of accuracy prediction in a limited amount of time. In addition, being the train images a representation of a Moon centered in the picture, the model would have learned always a constant position of (512, 512) during the training step. Therefore, it has been needed to modify the images, introducing a shift of the center of the Moon and a resizing of the image itself to a proper dimension. As such, all the known values of the center coordinates and the

radius seen in the image needed to be transformed accordingly to the new output image. A padding and a resizing have been therefore applied.

4.3.1 Data preparation

Given an image of the Moon in 1024×1024, a bounding box that contours only the illuminated surface is applied. To properly achieve a good estimation of the bounding box, the image has been firstly *binarized* and then a *closure algorithm* has been applied. This has been necessary due to the presence of some small portions of illuminated pixels that could act as outliers in the binarized image, letting the bounding algorithm to retrieve more regions to contour. This is more true the more the image is poorly illuminated. In that case, a sparse distribution of illuminated pixels would have led to tens of spread bounding boxes. Subsequently, the image is cropped on the basis of the bounding output and the initial center coordinates are related to a new origin placed in the upper left corner of the cropped image.

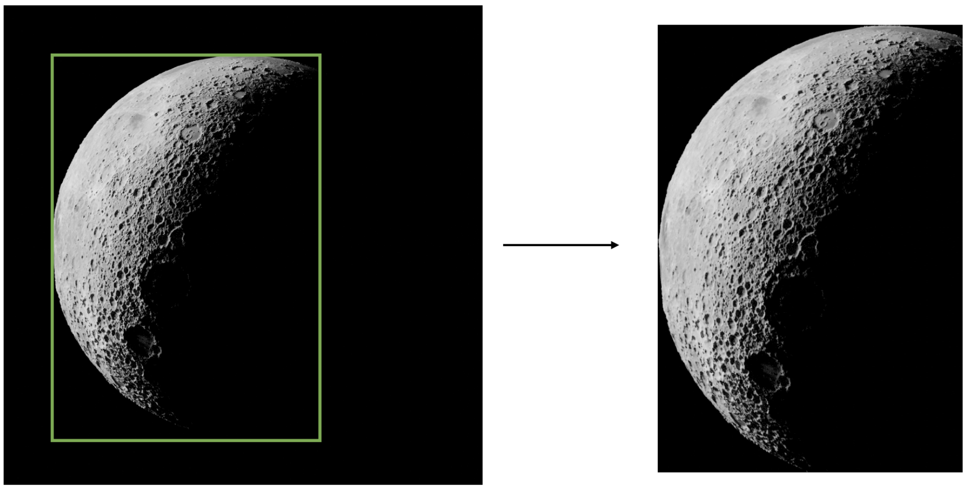


Figure 4.10: The image of the Moon is processed to retrieve the bounding box that contours the illuminated surface of the Moon. Then, the image is cropped accordingly to the bounding box.

After the image is cropped, to have different distribution of centers in the dataset, a random *padding* of 0 values is applied. Between the two sides of the bounding box, the longer one has been used to increase the sides to the nearest multiple of 128 (e.g. if the largest side is less than 256 but larger than 128, a padding is applied until a square image of 256×256 is reached[6]). This procedure allowed to rely on a random process that added rows or columns of zeroes in different positions, providing different distributions of centers.

[6]The same procedure has been applied for other possible combinations of sizes in the intervals 512-1024, 256-512, 128-256 and values smaller than 128.

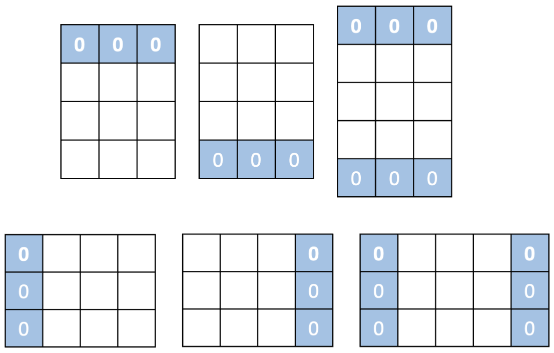


Figure 4.11: Given the image, a padding operation is performed. The position of the padding columns or rows is randomly chosen. This is applied on the top, on the bottom or in both positions until the desired size is reached. The same is randomly applied for the left and right positions.

Once the padded image is obtained, it is then resized as 128 × 128 pixels. Considering the scale factor obtained by diving 128 by the size of the padded image, the final radius $r_{resized}$ is immediately recovered as:

$$r_{resized} = r_{initial} \frac{128}{padded\ image\ size} \tag{4.3.1.1}$$

The Fig. 4.13 shows the the correctness of the transformations applied during the whole resizing problem, highlighting the new center of mass and the radius.

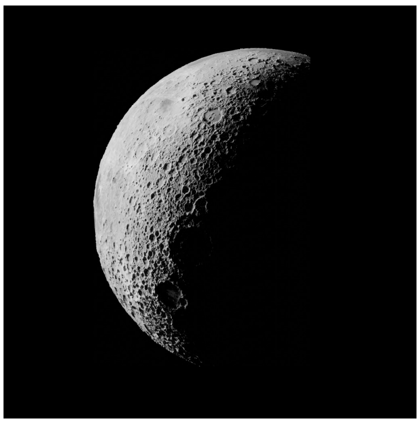


Figure 4.12: The padded image of the Moon. The initial picture of the Moon is processed to obtain a random shifted position of the center of mass of the satellite.

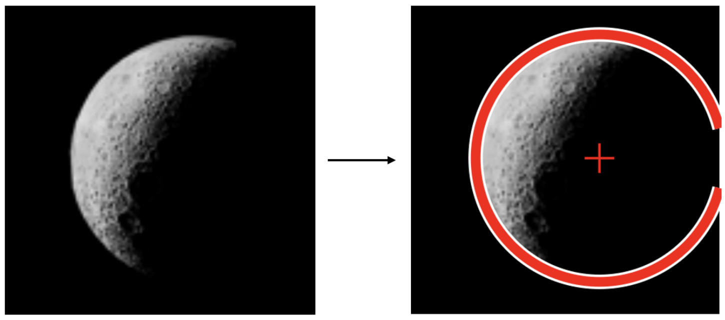


Figure 4.13: On the left: the resized image of the Moon to 128 × 128 pixels. On the right: the contour of the Moon and its center of mass are highlighted to check the correctness of the transformations applied both to the radius and to the center. It is possible to notice the reduction in quality of the image.

The transformation processing of the image has been applied to the entire dataset of 125090 images. Then, all the new values of the coordinates of the center and the value of the radius seen in the 128 squared pixel images have been added to the train datasheet. For what concerns the test set, in that case, the image processing required simply to scale the image to 128 × 128, by a factor of 8, and set the new coordinates center as (64, 64). The output value of the CNN for the radius, will be simply multiplied by 8 to retrieve finally the range estimation.

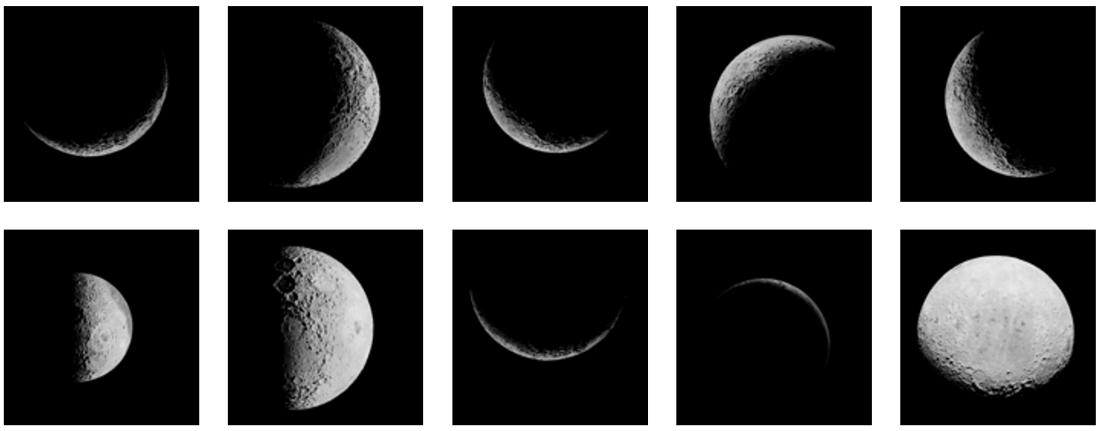


Figure 4.14: Some samples of resized and shifted images of the Moon after cropping and padding process.

4.3.2 Network implementation

Inspired by the architecture of the MarsNet network, a different model has been developed to better suit the environment of this work, still aiming to good performances and fast training.
The input image of size 128 × 128 is input to a first convolution layer that performs:

- a 2D convolution, with padding 1 × 1 and stride 1 × 1 with a kernel of size 3 × 3, receiving 1 channel and outputting 16 channels;

- an activation via Leaky-ReLU, with slope $\alpha = 0.01$;
- a Batch-Normalization with $\epsilon = 0.1$ and *momentum* 0.1;

A *max pooling* is then applied, with a kernel dimension of 2×2 that helped in halving the *width* and the *height* from 128 to 64. It has been preferred as method for dimension halving to a convolution step with stride 2×2, since, overall it performed better.
After this first convolution layer, a block of ResNet is applied. It is composed by:

- a 2D convolution with kernel size 3×3, stride 1×1, and padding 1×1;
- an activation Leaky-ReLU with slope $\alpha = 0.01$;
- a Batch-Normalization with $\epsilon = 0.1$ and *momentum* 0.1.

All the three points are repeated twice in the ResNet block. Then, the output of the ResNet block is added to the one of the initial convolution layer and passed to a second convolution block that performs a size halving of the volume thanks to a max-pooling operation. Overall, the ResNet + Conv2D + MaxPool block is repeated 4 times in the network. As such, after each pooling, the depth is doubled and the width and the depth halved.
After the data have been processed by the fourth max-pool, the final volume of $256 \times 4 \times 4$ is flattened into 4096×1. Subsequently, a dropout operation with probability $p = 0.2$ is applied, since it was observed during training it helped in guaranteeing a better convergence, preventing the overfitting. The values are the linearized to provide finally 3 output and activated by a Leaky-ReLU.

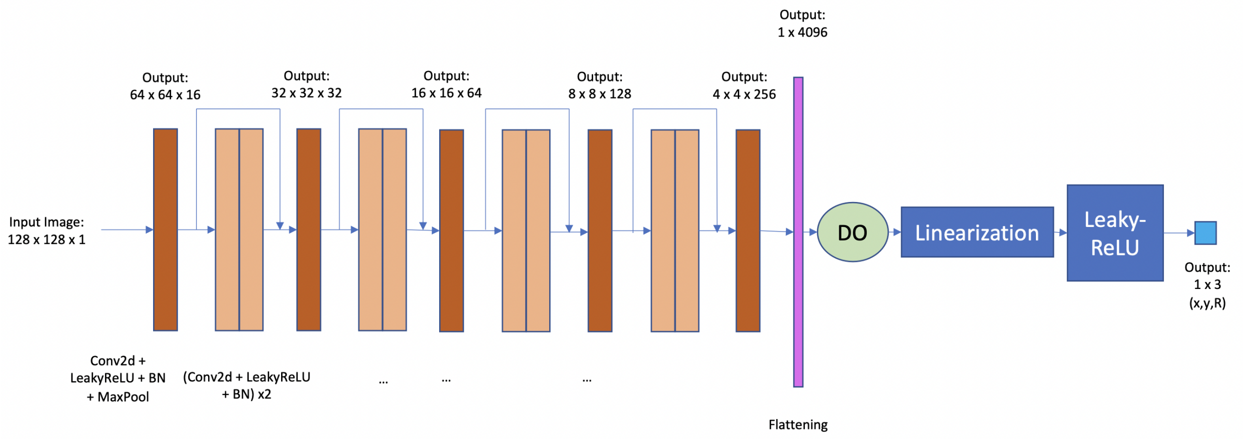


Figure 4.15: A representation of the ResNet for the regression task.

An optimal learning rate has been found to be $lr = 0.001$ during the training as trade-off between speed and accuracy. Between Adam and SGD optimizer, the SGD has been chosen, which provided overall a better learning with a momentum $= 0.9$. The Huber loss has been kept from the original model, providing overall a better convergence and a suitable solution for the architecture involved. In addition, with such this structure, the model worked with

$798,723$ trainable parameters.

The network has been developed and trained by making use of *PyTorch*[7] framework in *Google Colaboratory*[8], which allowed to the parallel computations provided by the GPUs.

4.3.3 Training phase

Despite the 125090 image data rendered for the training set, it has been necessary to finally define a smaller batch of images, mainly to operate in the time limits defined by Google Colab's GPUs. Therefore, a trade-off has been carried out to identify the best number of data needed to perform an accurate but still fast training along with suitable data samples. In this case, a training set of 34940 images has been used for the learning. Each of the images came along with three labels, namely the two coordinates of the center of mass, and the radius in pixels of the Moon in the image that were used by the network for the learning process.
In addition, for the training process and parameter tuning, a validation set has been created, containing the 10% of the training set. Moreover, the training set was shuffled in the batches, allowing to train the network on different type of images, involving different possible moments around the Moon.
During the training, the hyperparameters of the network, along with the optimizers, have been changed with respect to the original *MarsNet* configuration. Firstly, the Stochastic Gradient Descent optimizer has been chosen, with a momentum of 0.9. During the training it was observed that it allowed to generalize slightly better than the adaptive Adam optimizer. Additionally, to train the model within an acceptable amount of time, providing a good learning, after some iterations, and making a trade-off between time costs/accuracy, a batch size of 200 images has been finally chosen, trained for 100 epochs. The optimal learning rate that allowed an accurate convergence has been chosen finally to be equal to 0.001, and the Huber loss function has been used like in the original model. The train, validation and test accuracy have been evaluated by making use of the Mean Squared Error, suitable for a regression process

Class imbalance

The application of the random padding along with the resizing, created a dataset of images in which high values of radius of the Moon were preferred. This was mainly due to the random and not controlled nature of the padding method applied to bounding box of the Moon, rather than to a slightly extended area. As such a class imbalance derived from the the final rendering, with the labels of the radius skewed more to values larger the 40. As such, it has been needed to modify the train set, partitioning it in 70% of the images with shifted center of mass, and 30% of only resized images. In this way, the network could be capable to train also on smaller values of radius.

[7]https://pytorch.org . Last accessed: 22-11-2021
[8]https://colab.research.google.com . Last accessed: 22-11-2021

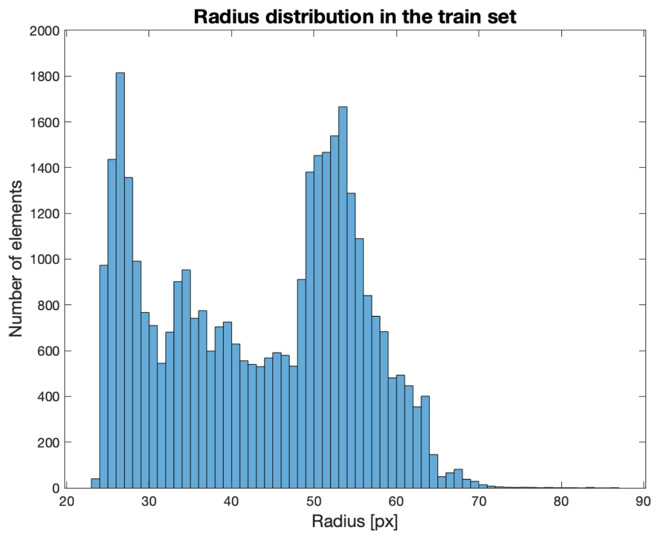


Figure 4.16: The balanced train set. The radius values ranges from about 20 pixels up to more than 60.

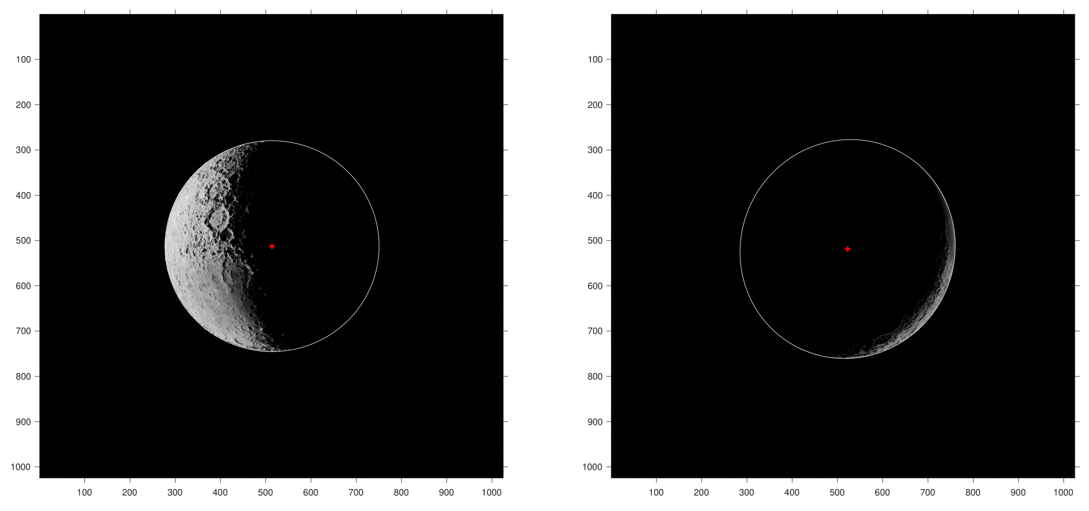
Chapter 5

Results

The following chapter will present the results that have been obtained by performing the two algorithms for the center and apparent diameter finding on the images of the Moon. As comparison data, the test set of 8935 images has been used for the evaluations.
The computations have been carried out on MatLab R2020b for the standard CAD algorithm and ellipse fitting and on Google Colaboratory for the neural network training.

5.1 Standard CAD

The ellipse fitting based CAD algorithm provided interesting and accurate solutions in different illumination conditions and at different LUMIO-Moon ranges. As it can be observed, the approach dealt with good accuracy the distinction between the lit horizon and the terminator, allowing to recover the fictitious direction of the sun in the image plane and to reconstruct the missing part of the ellipse, even in extreme conditions with little illuminated surface.



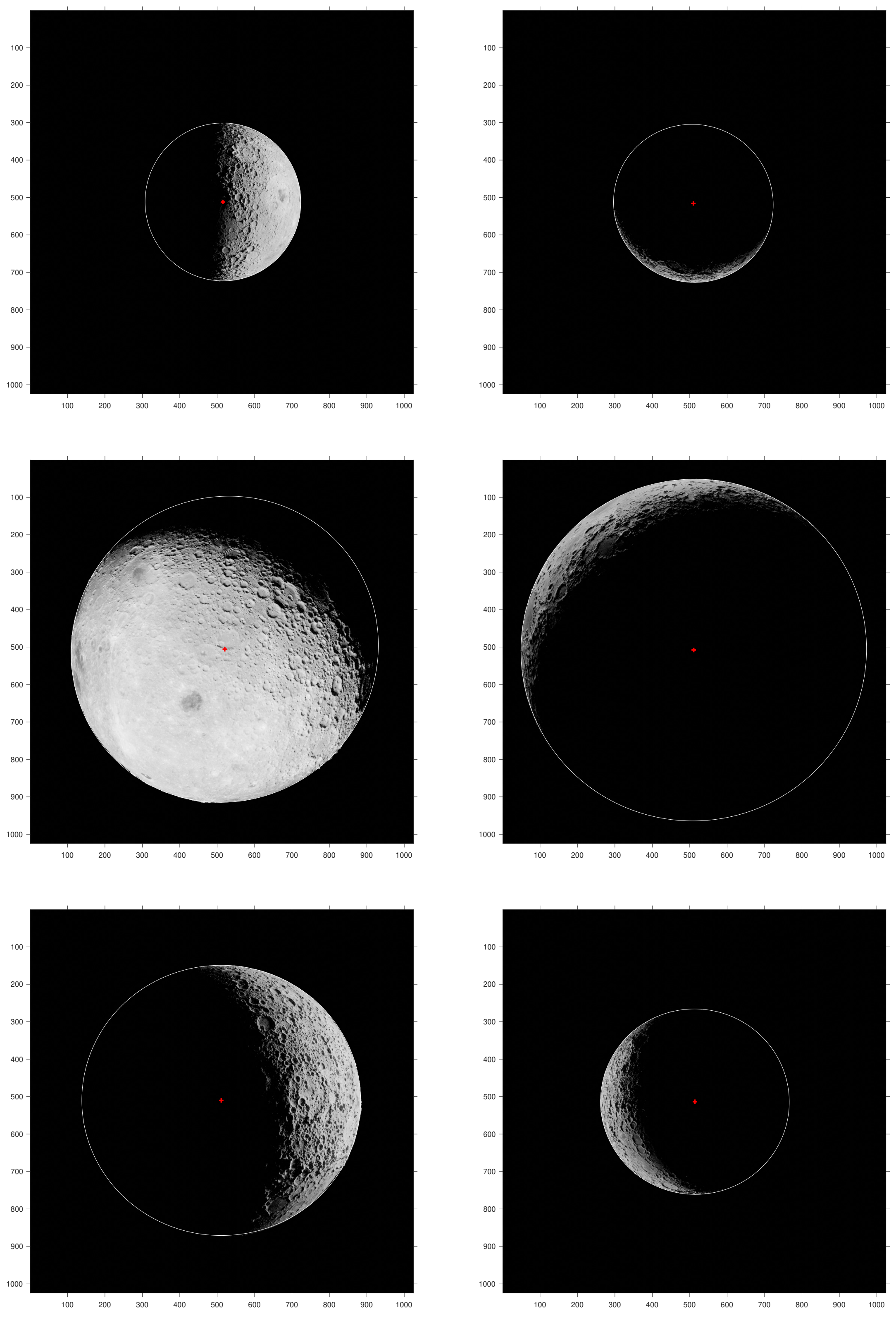


Figure 5.1: Samples of the images of the Moon processed by the standard image processing CAD. The algorithm processes the images in every condition of illumination, estimating the ellipse highlighted in white and the center of mass of the Moon.

In particular it is possible to observe how well the algorithm estimated the position of the center and the radius of the scanned Moon by looking at the graph below.

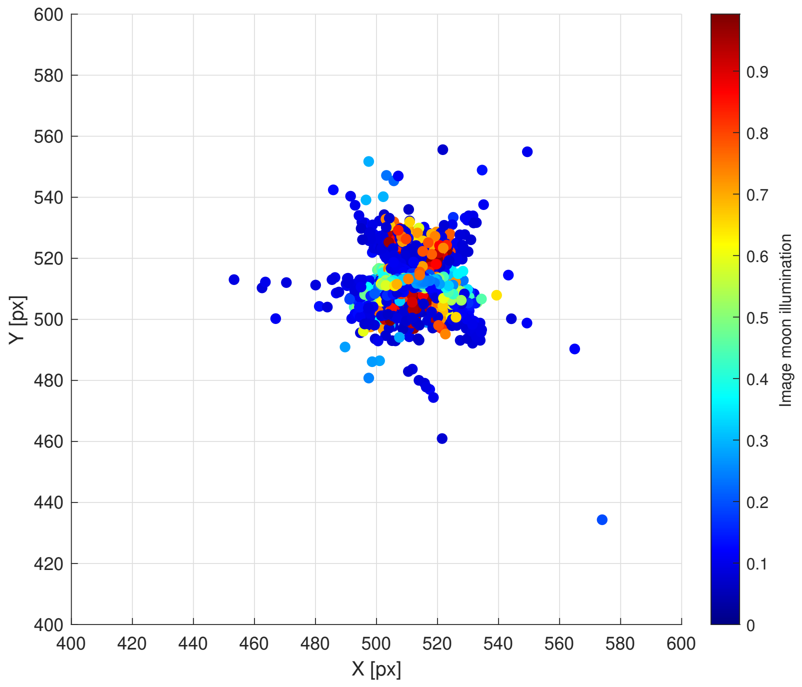
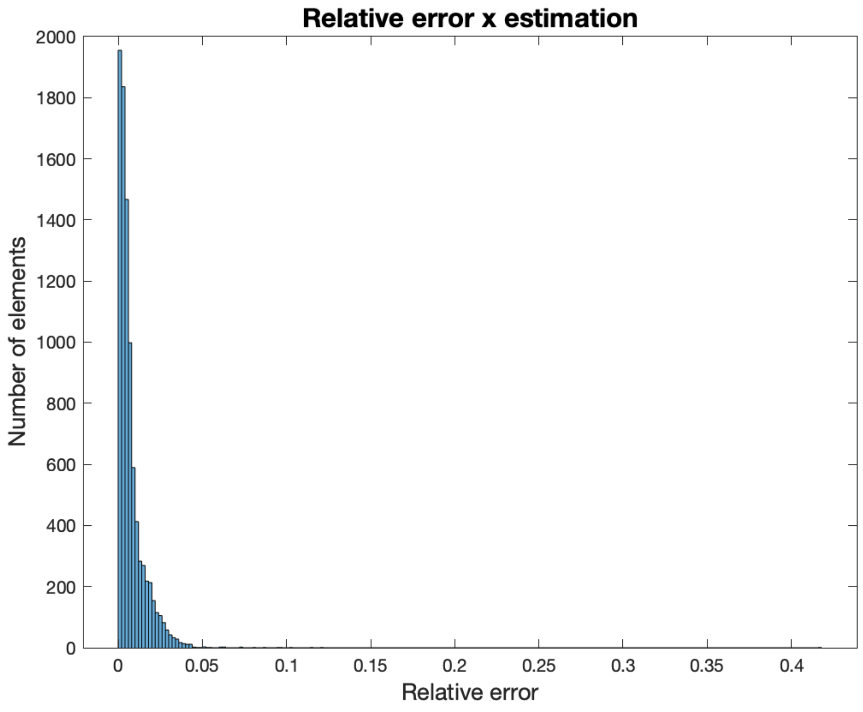


Figure 5.2: The estimated position of the center of mass of the whole image test set. The x and y coordinates are plotted considering the amount of surface illumination.

It is immediately noticed that the algorithm performed not perfectly when the illumination condition given by ρ is low for the center estimation.
As stated previously in the *eigenvalue analysis* section 4.2.1 the illumination condition was computed by using the ratio of the eigenvalues of the problem. For values near to 0, the surface of the Moon is largely shadowed, providing the algorithm a smaller arc on which fit the ellipse. However, by looking at the Fig. 5.4, it is worth noticing that the radius estimation performed well even for the low light conditions. Indeed, almost all the points estimated independently from the illumination condition, form a linear behaviour with respect to the real radius value.



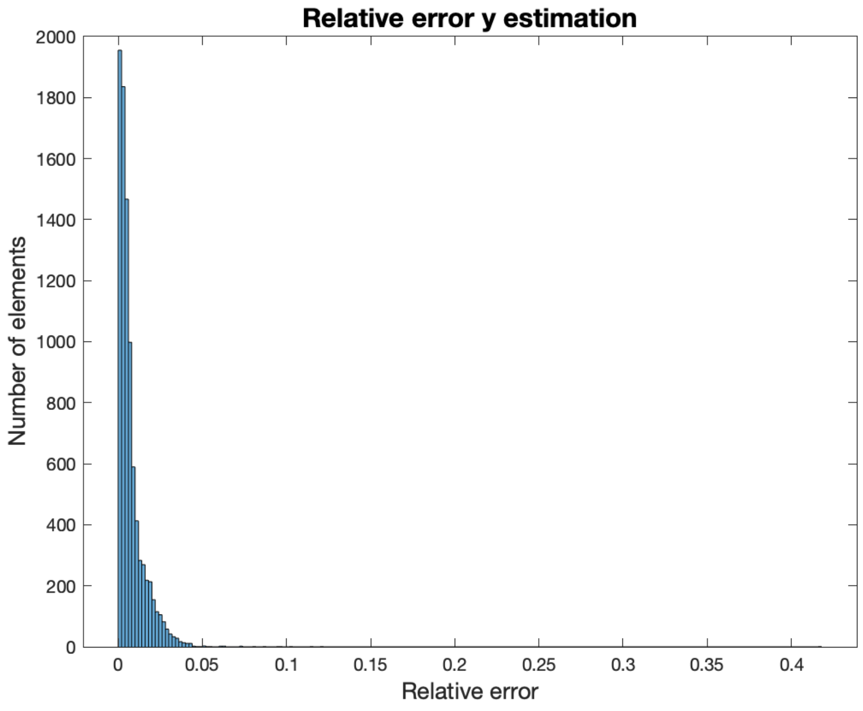


Figure 5.3: On top: the relative error distribution for the x coordinates; On bottom: the relative error distribution for the y coordinates for the ellipse fitting method.

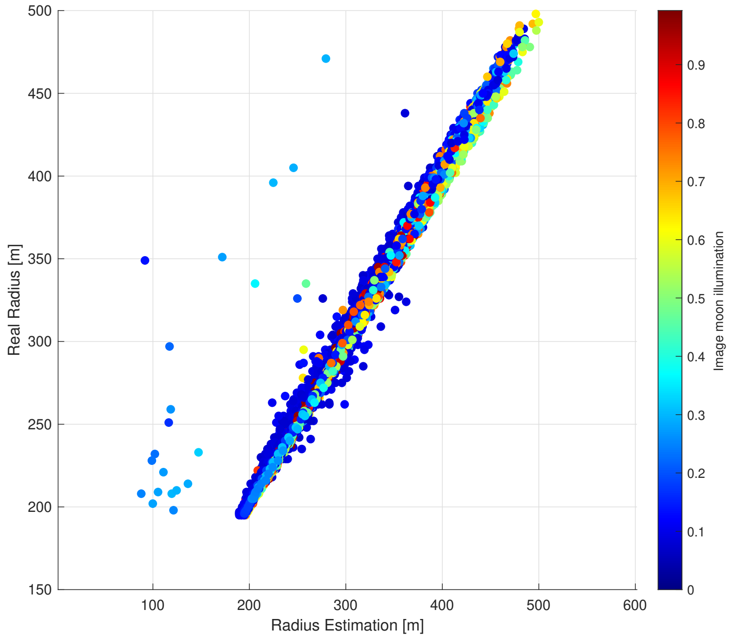


Figure 5.4: The estimated radius is plotted against the real values of the radius of the Moon in the image. The estimations are influenced by the ρ illumination parameter.

As such, the range estimation is straightforward by making use of the pinhole model relation (Eq. (4.2.1.5)).

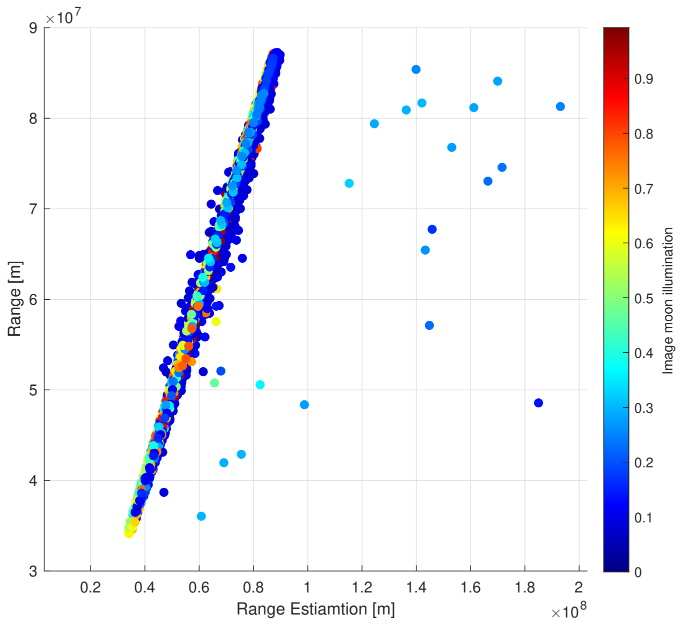


Figure 5.5: The estimated range is plotted against the real values of the radius of the Moon in the image. The estimations are influenced by the ρ illumination parameter.

As expected, the linear behaviour was certainly respected. Overall, the model succeeded in the ellipse fitting even for extremely poor illuminated images. An example is shown here below:

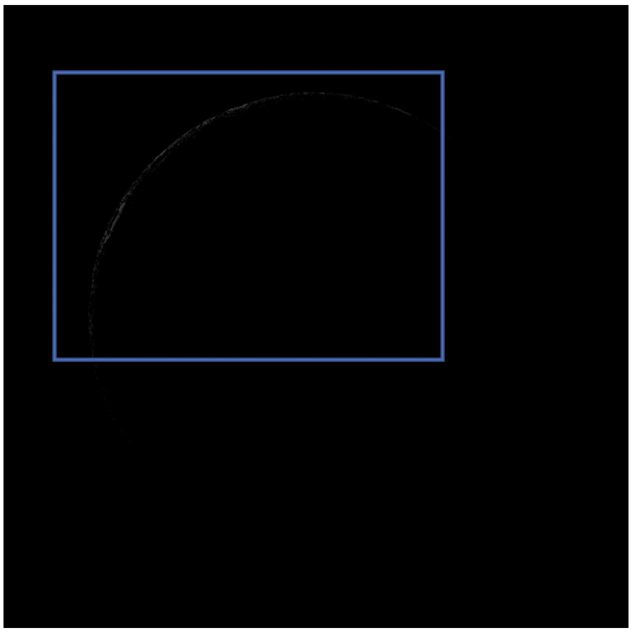


Figure 5.6: An image of the Moon in extreme poor illumination conditions. A blue box is used to highlight the illuminated arc of the horizon

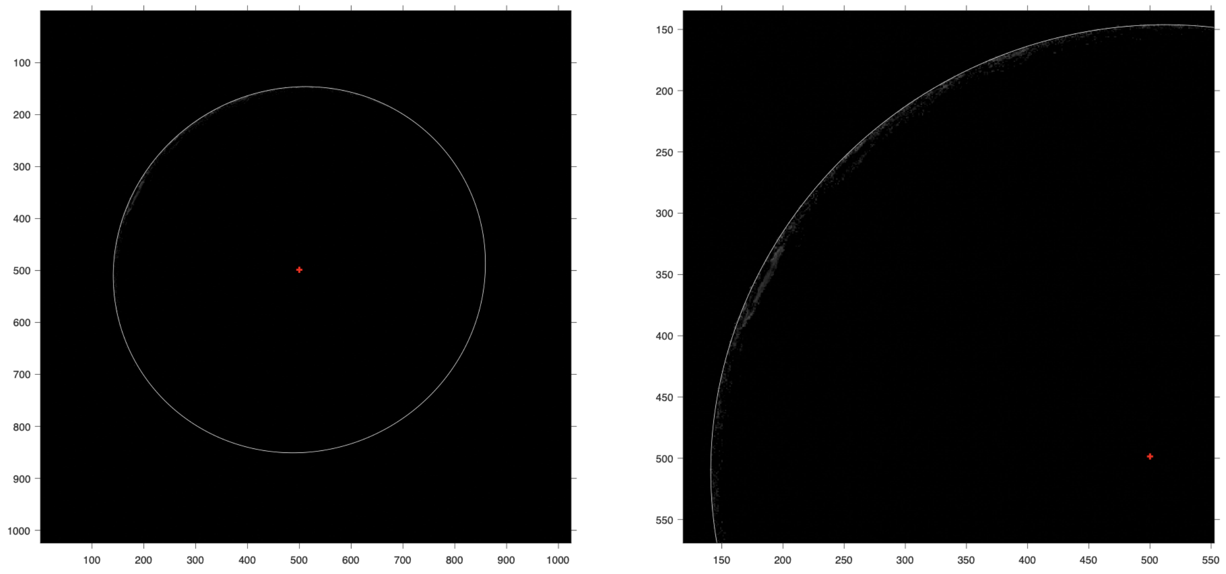


Figure 5.7: The same image of the Moon is processed by the CAD algorithm and the ellipse is fitted. On the right a zoomed view of the same image is presented for better visualization. A radius of 355 pixels is predicted against the real 370 pixels.

Although the algorithm succeeded in retrieving an ellipse from low illuminated surfaces, it could not deal with images where no pixels were illuminated. In particular, 26 images of data set did not provide any illuminated surface. For those cases, the binarization algorithm could no be applied and, as such, the image processing not performed. Among these samples, there are some poor illuminated images where, even though the method succeeded in the binarization step, the high sparsity of the points did not allow to retrieve an accurate prediction of the ellipse.

The points in the Fig. 5.5 that stand outside from the average behaviour, are instead due to really small illuminated arc that could no be representative of the real size of the Moon.

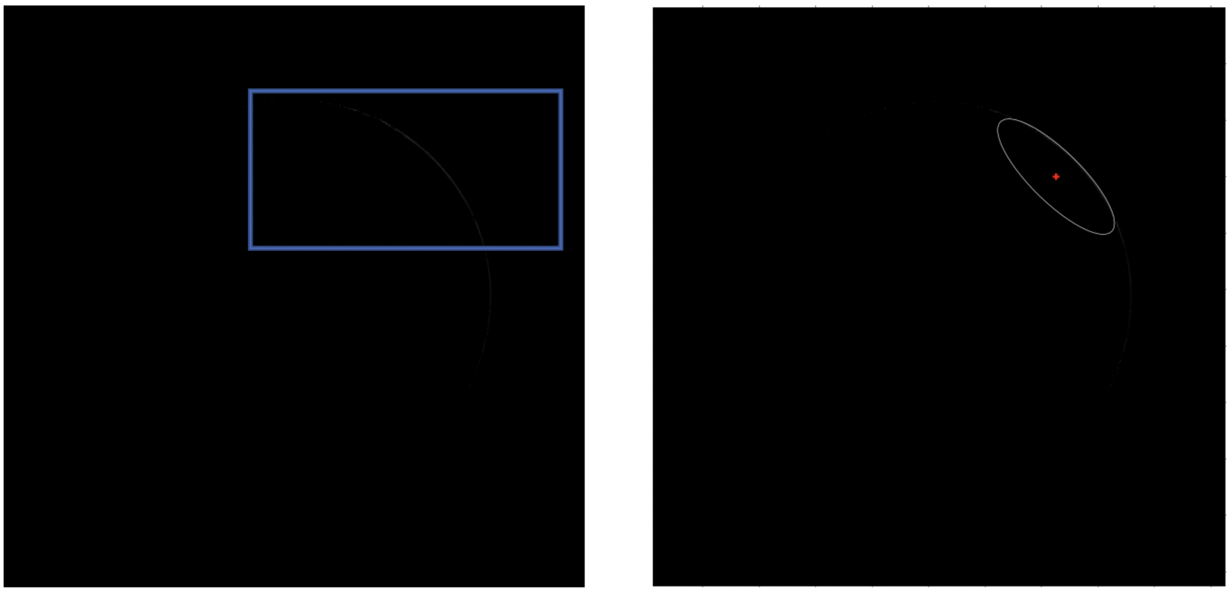


Figure 5.8: An example of poorly illuminated surface. the illuminated arc is not representative of the real radius of the Moon in the picture. A blue box is used to highlight the illuminated arc. On the right, the ellipse fitting is applied.

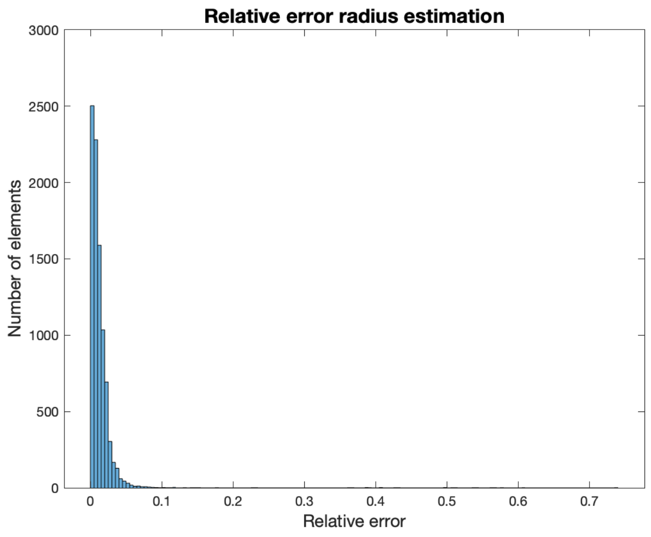


Figure 5.9: The distribution of relative errors for the radii estimation from ellipse fitting.

Overall, the CAD image processing method provided on average a relative error for the x coordinates equals to about 1.04% and 0.78% for the y coordinates. Instead a mean relative error of 1.60% has been obtained for the radius estimation. For what concerns the range estimation, the average relative error has been found to be equal to 1.70%.
The highest relative error for the radius has been obtained for the Moon in the Fig. 5.8, equals to 76%. For the same image the worst relative error for both the x and y coordinates estimations was equal to 42% and 280% for the range estimation of the same picture. Moreover, the Fig. 5.9 shows how the errors are more skewed towards low values. The same is observed for the coordinates estimations.

	X	**Y**	**Radius**	**Range**
Mean relative error	1.04%	0.78%	1.60%	1.70%
Highest relative error	42%	42%	76%	280%

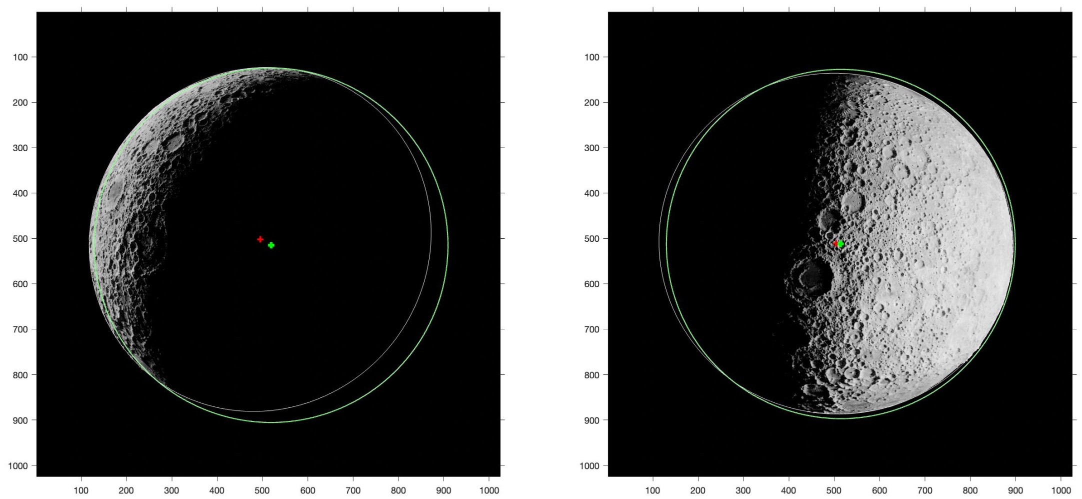
Table 5.1: Ellipse fitting relative errors

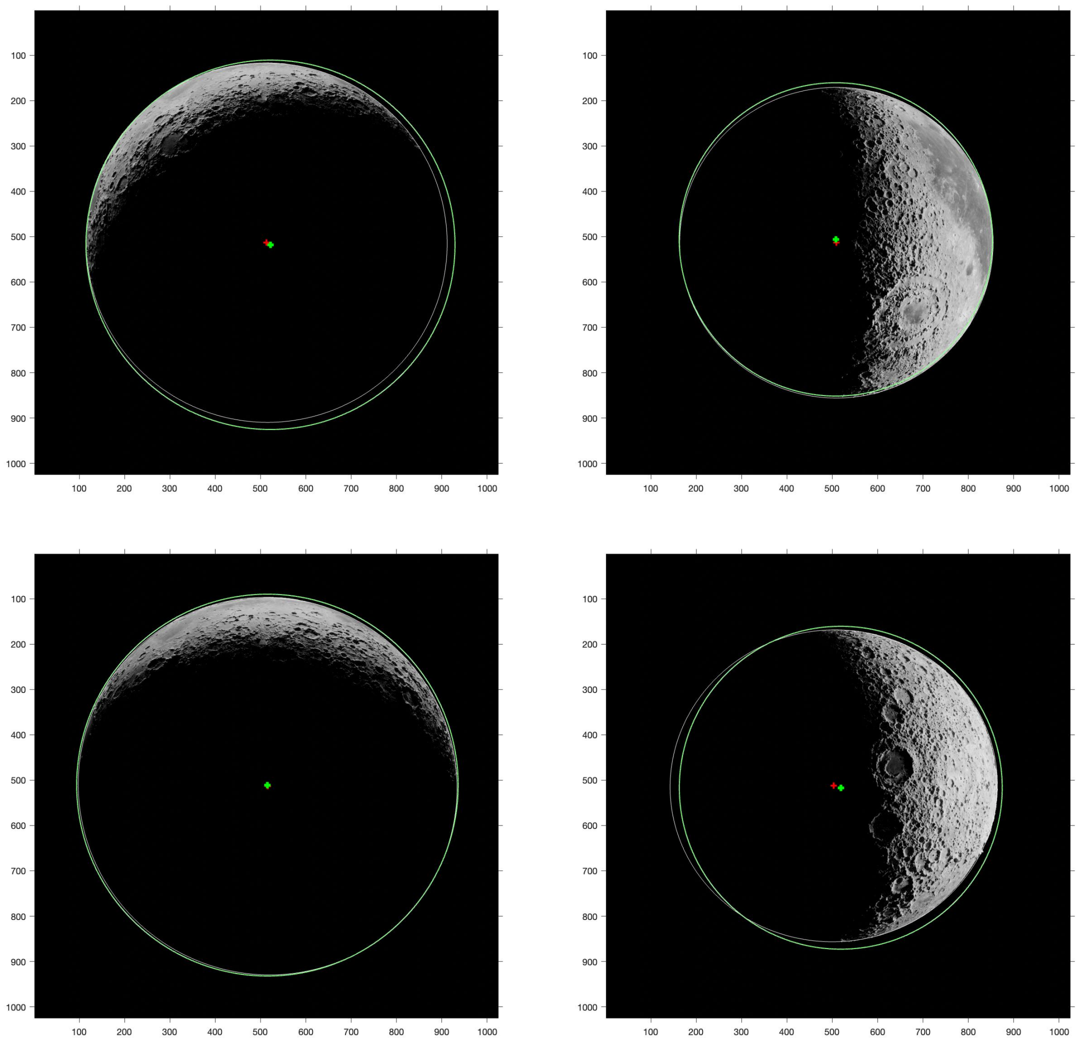
The method allowed to compute the range, the center of mass and the radius of the Moon with a good estimation error. The images where no surfaces were presented or with not representative arcs of the horizon, acted as the main elements for poor estimations.

5.2 Convolutional Neural Network

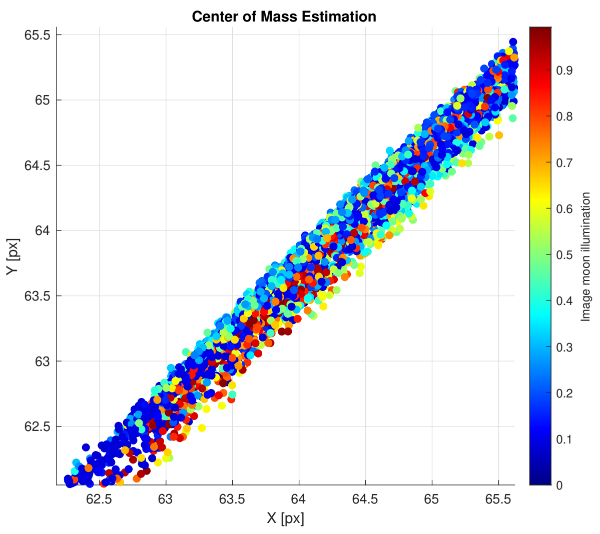
The model explained previously in section 4.3.2, has been here applied for the deep learning approach. In particular, the training has been carried out by making use of 34940 images of 128 × 128 pixels. However, to show a comparison with the standard CAD image processing, the figures below will present how the two algorithms performed on the same image, after scale transformation to 1024 × 1024 pixel size.

Figure 5.10: Some samples of Moon images in 1024 × 1024 pixels. In green the output of the CNN, in white the output of ellipse fitting method.





Even though the network presented interesting and accurate outputs, like the ones presented here above, overall it related the x and y coordinates with a linear behaviour. As such, the y predictions seems to increase with the values of the correspondent x.
However, the mean relative error for the x and y coordinates predictions are respectively equal to 2.31% and 2.26%. Indeed, even though the model related x and y linearly, it is worth noticing that only the 7% of the whole test set has been predicted to have a center of mass standing 5 pixels away from $(64, 64)$, for both x and y directions, corresponding to a 6% of relative error.
This can be observed in the Fig 5.1: in the upper image, a dense representation of points has been estimated to lie near (64, 64) with an error less than 6%; moving far from the ground truth values, the predictions are more sparse, representing only the 7% of the whole test set.
Moreover, it is possible to notice that the network dealt the images almost independently from the illumination condition, not skewing better estimation neither for high ρ nor for low ρ values.



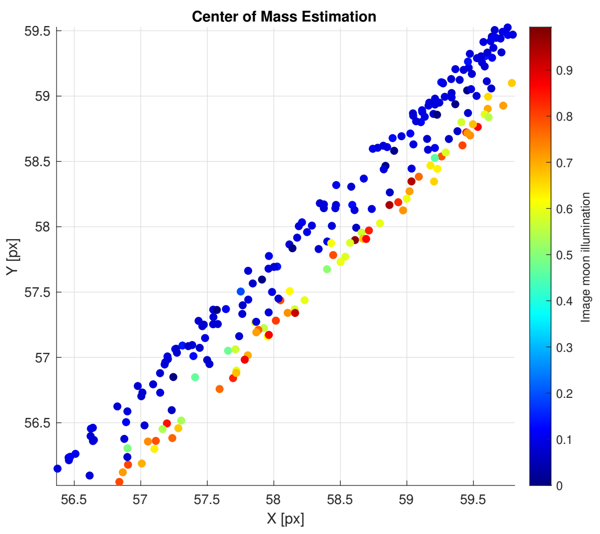
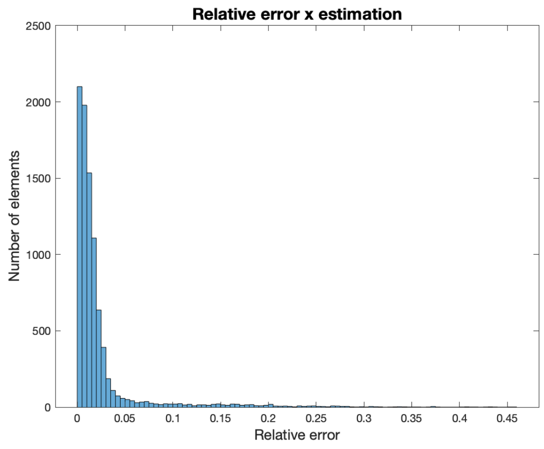


Figure 5.11: On top: the x estimations are plotted against the y estimations. The colorbar defines the illumination conditions; on bottom: a representation of the estimated points for relative errors larger than 6%.

The low value of relative errors obtained for the center estimation are presented in the histograms in Fig. 5.12: the majority of the errors are obtained for values less than the 5% for both the coordinates.



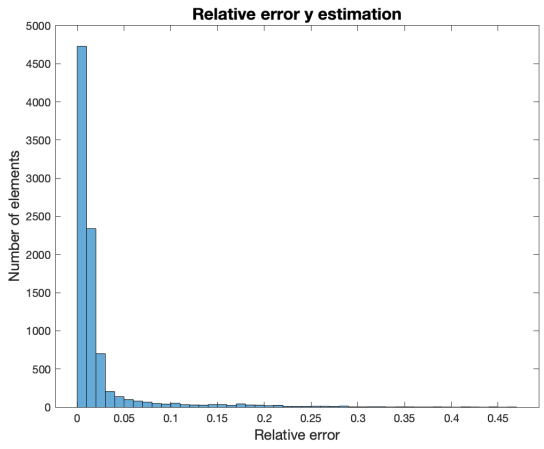


Figure 5.12: On top: the relative error distribution for the x coordinates; On bottom: the relative error distribution for the y coordinates.

For what concerns the radius estimation, the model performed worse than the center of mass evaluation. Indeed, an average relative error of about 25% has been obtained. Even though the figure shows a poor linear behaviour for different samples, it outputs not accurate estimations for some radii. In particular, the model was prone to overestimate the radii values. In addition, it is also possible to notice that the model failed for the majority to estimate small values of radii.
The Fig. 5.15 shows, indeed, how the relative errors for the radii estimation are present in large quantities for values larger then 10%.

	X	**Y**	**Radius**	**Range**
Mean relative error	2.31%	2.26%	25%	18.87%
Highest relative error	45.53%	46.84%	81%	68.25%

Table 5.2: Ellipse fitting relative errors for CNN.

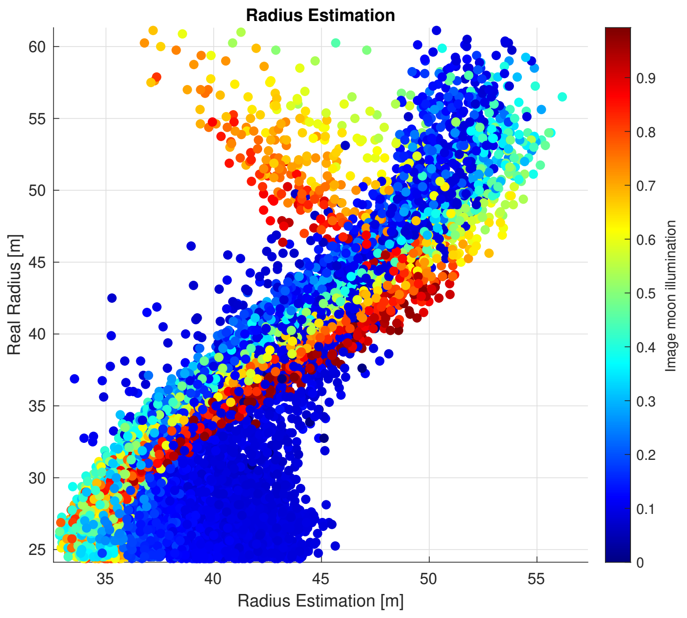


Figure 5.13: The estimated radii are plotted against the real values considering the illumination condition.

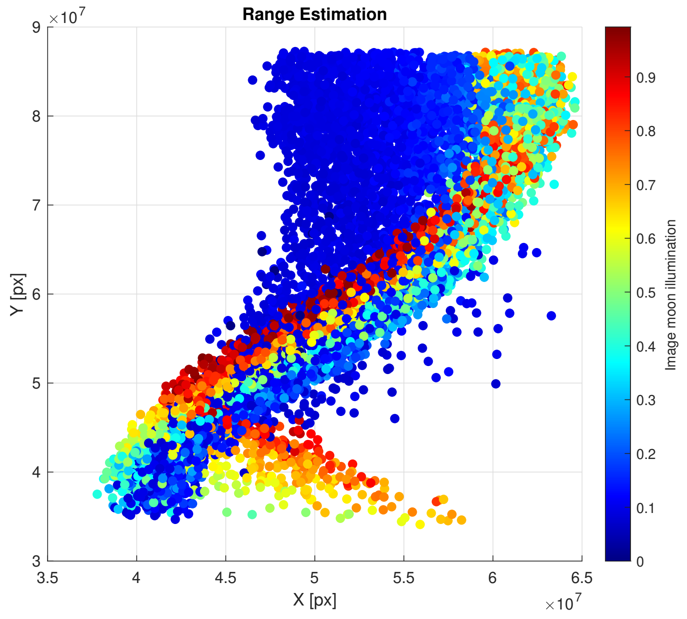


Figure 5.14: The estimated ranges are plotted against the real values considering the illumination condition.

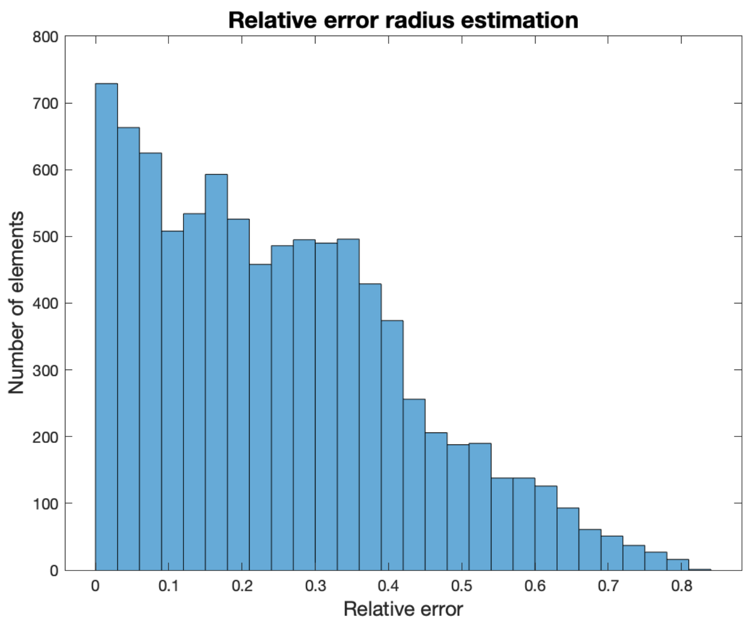


Figure 5.15: The distribution of relative errors for the radii estimation.

The training of the network has been performed over 100 epochs considering a batch size of 200 images. The model did not show overfit, presenting a lower value for both test accuracy and test loss. The values for the training, validation and test set are reported here in the Tab. 5.3:

Training loss	Validation loss	Validation accuracy [MSE]
6.13	6.05	93.8

Table 5.3: Final loss and accuracy values.

Test loss	Test accuracy [MSE]
3.35	34.23

Table 5.4: Final loss and accuracy values for test set.

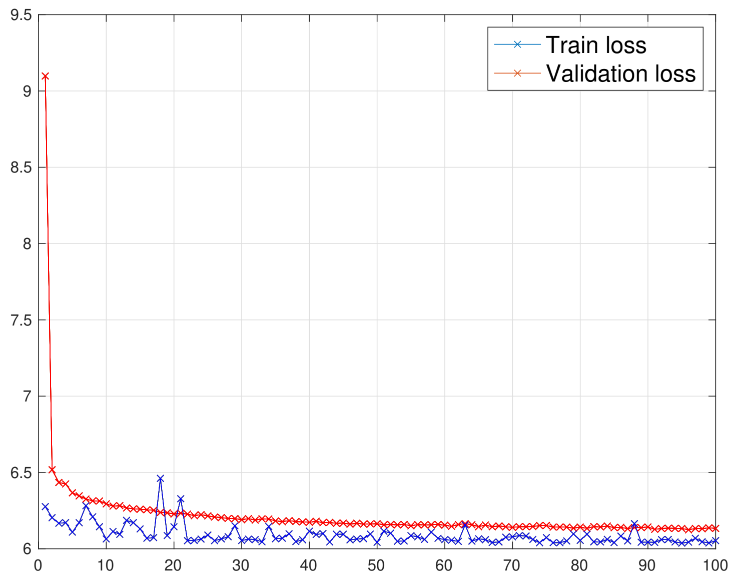


Figure 5.16: The train loss and the validation loss (Huber loss) for 100 epochs. Batch size of 200 images; SGD optimizer; lr=0.001, momentum=0.9

Chapter 6

Conclusions and Future Work

The aim of this work was to understand and verify if a *machine learning* technique, namely a convolutional neural network, could outperform a consolidated and highly space proven centroiding technique. To achieve the results, the reader has been guided during the path, starting from a general survey of the state of the art of image processing in the space environment. The different methods have been presented to provide an idea of what image processing means for autonomous navigation, historically travelling from the initial approaches of Mariner IX up to the last machine learning based methods. In the chapter 2, some analytical insights have been presented for different methods, highlighting in particular the CAD method based on ellipse fitting on which this work relied on. Then, through the chapter 3, the techniques and the approaches used in machine learning have been detailed, showing the *MarsNet* approach as centroiding technique. Then in chapter 4 and in chapter 5 the methods have been applied and finally compared, trying to understand which was the method that dealt better with centroiding, in different illumination conditions.

The world of machine learning and image processing is one of the most attractive and interesting in computer science and specifically in computer vision, in the actual technological era. The current work allowed to delve into innovative techniques, understanding the insights behind them through a deep research over the initial months. In this way it has been possible to discover different methods, learning their advantages and disadvantages, depending on the aim of the work, and developing idea and intuitions. Numerous have been the solutions that have been studied, and different the capabilities earned during the path, delving into *Python* and its packages concerning about image processing like *OpenCV* or the Image Processing Toolbox for MatLab. At the same time, it has been possible to work and specialize in *dataframes* using *pandas* and *csv* modules. The rendering methods presented a deep and fascinating world that presents room for innovations and more understanding, allowing to discover softwares like Blender, delving into its automation processes.
Along with it, the *PyTorch* framework and the *Colab* tool allowed to investigate the better solutions for dealing with artificial intelligence, in numerous and manifold ways.

6.1 Conclusions

As it has been possible to notice from the results presented in the chapter 5, the center and apparent diameter finding based on the ellipse fitting method succeeded in its task,

confirming to be well proven. In particular, it worked with high accuracy, trying to retrieve the center and the fitting ellipse in many different illumination conditions.
Overall, the illumination of the surface of the Moon was fundamental for the center estimation. As observed, the large deviations from the real CoM happened mainly for low light condition, presenting a sparsity of centers but still with low relative error values. On the other side, the illumination conditions did non influenced as much the radius and range distribution. The linear relation that has been found between the estimated values and the real ones confirmed the success of the algorithm for the radius prediction task, independently from the ρ values that where almost perfectly distributed. To confirm this trend, the histograms for the relative errors distribution showed a shift toward really small values.
For poor illuminated image, fundamental was the distribution of the illuminated pixels over the arc, particularly when few shadows (like the ones due to craters) were displayed along the terminator. In that case even for hard conditions, the model provided interesting solutions. However, it has been possible to notice that, for very low illumination conditions, the small arcs from the lit horizon were not sufficient for an accurate estimation.
Overall, the ellipse fitting allowed to reach really low values of relative errors, proving its success in the task.

On the other hand, the Convolutional Neural Network model, did not always provide the expected results. Even though it succeeded in retrieving the needed information from the image in different cases (e.g. Fig. 10), overall, it was highly biased by the distribution of values in the target labels, highly influenced by the the generally high modules of the coordinates centers. However, it has been possible to notice during training how this bias could be reduced increasing the number of images for the train, coming at the cost of a time expensive solutions that was not viable.
The method, as such, worked well on the coordinates estimation, not being influenced by the illumination condition. Indeed in Fig. 5.11, it was possible to notice a good distribution of illumination conditions along the predictions. The model performed almost a linear relation between the two coordinates. However, the prediction were still good and accurate, providing an estimation error of about 2% for x and y. The error distributions, indeed, showed how the model succeeded overall in predicting the coordinates of the center, not outperforming, however, the ellipse fitting algorithm.
Differently were the solutions for the radius estimation and for the range. As such, in that case, an almost linear and general behaviour between the estimated and real values was defined. In particular, the model was more prone to over estimate the solutions for the radius, and this was particularly true for the low illumination condition. Indeed, the worst accuracy in terms of relative error came for the radius and range estimations.
However, the model that was found during the training to perform overall better, was the one previously presented in section 4.3. It was obtained by recursively changing the parameters, the layers and the functions involved. Indeed, to achieve a fast training, along with a good accuracy, the structure has been changed to be adapted to the problem. As such, the max-pooling operations have been preferred to a convolution with stride 2×2, and a complete convolution layer involving both activation a batch-normalization used starting from the very beginning of the network. The ResNet block where reduced to four and a dropout with probability of 0.2 was placed only after the flattened output.

Overall, the ellipse fitting model succeeded in its task performing with high accuracy in different illumination conditions, proving its stability in image processing environment, outperforming the convolutional neural network.

6.2 Future works

The work that has been carried out up to now, showed the weaknesses and the strengths of both the two models. The main problem with the ellipse fitting method was found to be the approach to poorly illuminated images of the Moon.
In the case of sparse illuminated pixels, the ellipse fitting method did not properly succeeded, particularly when the identified arc was too short.
Therefore, a refinement of the model could be achieved by studying different approaches that could highlight sparse pixels that are zeroed during the threshold and binarization process. In this way it would be possible retrieve more pixels belonging to the same arc of the horizon, which may provide a better estimation for the CAD.
At the same time, even other approaches could be carried out, based on different ellipse fitting methods. Different datasets could be also generated, introducing distortion phenomena due to lenses to emulate an even more realistic scenario.

For what concerns the convolutional neural network, the solutions are even more and manifold. Different are models that could be used for performing a multi-output regression task.
The main weakness observed for the deep learning approach, has been that it was prone to overestimate the radius values. It was observed, however, that the training was more accurate the more the training dataset was large in number of items. However, such a approach should be observed with care, depending on the amount of resources that are available, like GPUs and frameworks used for the model building. In that case, it would be recommended to apply different solutions for augmenting data to shift the center of mass. The random nature of the padding and resizing method influenced the output, by preferring a final size of the radius that is, on average, shifted towards high values, neglecting images of the moon at high distances. This could be also due to the choice of the bounding box of the illuminated surface of the moon as starting point of the padding.
As such, apart from this solution, it could be suggested to carry out a random padding starting from a random variation in the bounding box's sizes. Therefore, it would be possible to build an image starting from a high bounding surface, helping in reaching a more manifold dataset.
The data orientation of this approach, could be also compared with a solution in which a normalization of the outputs is carried, reducing the influence of the center of mass coordinates on the radius.

Therefore, the solution that deep learning presents are uncountable. The approach could be more model oriented, where the optimization of the model is highly preferred to a large amount of different data. In that case, it could be possible to work on the different parameters involved, changing the learning rate, on the basis of *lr finders* and *Fit-1-Cycle* approach, or even working with different optimizers, coming up with a better solution.

The world of image processing, using standard approaches or deep learing ones, presents uncountable solutions, providing room for improvements and space for innovation.

Bibliography

[1] John A. Christian. “Accurate Planetary Limb Localization for Image-Based Spacecraft Navigation”. In: *Journal of Spacecraft and Rockets* 54.3 (May 2017), pp. 708–730. DOI: 10.2514/1.A33692.

[2] John A. Christian and Glenn Lightsey. “An On-Board Image Processing Algorithm for a Spacecraft Optical Navigation Sensor System”. In: *Journal of Spacecraft and Rockets* 49.2 (Mar. 2012), pp. 337–352. DOI: 10.2514/1.A32065.

[3] Navin Jerath and Hiroshi Ohtakay. “Mariner IX Optical Navigation Using Mars Lit Limb”. In: *Journal of Spacecraft and Rockets* 11 (1974), pp. 505–511. DOI: 10.2514/3.62114.

[4] Campbell J., Synnott S., and Bierman G. “Voyager Orbit Determination at Jupiter”. In: *IEEE Transactions on Automatic Control* 28.3 (1983), pp. 256–268. DOI: 10.1109/TAC.1983.1103223.

[5] Hoffman A., Green N., and Garrett H. “Assessment of In-Flight Anomalies of Long Life Outer Planet Missions”. In: *Proceedings of the 5th International Symposium on Environmental Testing for Space Programmes* (2004).

[6] M. Rayman et al. “Results from the Deep Space 1 Technology Validation Mission”. In: *Acta Astronautica* 47.2 (2000), pp. 475–487. DOI: 10.1016/S0094-5765(00)00087-4.

[7] W. Owen, P. Dumont, and C. Jackman. “Optical Navigation Preparations for New Horizons Pluto Flyby”. In: *Proceedings of the 23rd International Symposium on Space Flight Dynamics (ISSFD)* 47.2 (2012), pp. 475–487. DOI: 10.1016/S0094-5765(00)00087-4.

[8] A. Pellacani et al. “HERA Vision Based GNC and Autonomy”. In: *8th European Conference for Aeronautics and Space Sciences* (2019). DOI: 10.13009/EUCASS2019-39.

[9] J. Riedel, W. Owen, and J. Stuve. “Optical Navigation During the Voyager Neptune Encounter”. In: *American Institute of Aeronautics and Astronautics* (1990), pp. 118–128. DOI: 10.2514/6.1990-2877.

[10] David A. Lorenz et al. “Lessons learned from OSIRIS-REx autonomous navigation using natural feature tracking”. In: *2017 IEEE Aerospace Conference*. 2017, pp. 1–12. DOI: 10.1109/AERO.2017.7943684.

[11] Dae Min Cho et al. “Robust Feature Detection, Acquisition and Tracking for Relative Navigation in Space with a Known Target”. In: *Guidance, Navigation, and Control and Co-located Conferences* (Aug. 2013). DOI: .2514/6.2013-5197.

[12] Lena M. Downes, Ted J. Steiner, and Jonathan P. How. “Lunar Terrain Relative Navigation Using a Convolutional Neural Network for Visual Crater Detection”. In: *American Control Conference* (2020). DOI: 0.2514/6.2011-6490.

[13] Lena M. Downes, Ted J. Steiner, and Jonathan P. How. “Deep Learning Crater Detection for Lunar Terrain Relative Navigation”. In: *AIAA SciTech Forum* (Jan. 2020). DOI: 0.2514/6.2020-1838.

[14] Ari Silburt et al. *DeepMoon: Convolutional neural network trainer to identify moon craters.* May 2018.

[15] Tanner S. Campbell et al. “A deep learning approach for optical autonomous planetary relative terrain navigation”. In: 2017.

[16] Sumant Sharma and D’Amico Simone. “Neural Network-Based Pose Estimation for Noncooperative Spacecraft Rendezvous”. In: *IEEE Transactions on Aerospace and Electronic Systems* 56.6 (June 2020), pp. 4638–4658. DOI: 10.1109/TAES.2020.2999148.

[17] F. Topputo et al. “LUMIO: a Cubesat at Earth-Moon L2”. In: *The 4S Symposium 2018, Sorrento, Italy* (2018).

[18] V. Franzese, Di Lizia P., and Topputo F. “Autonomous Optical Navigation for the Lunar Meteoroid Impacts Observer”. In: *Journal of Guidance, Control, and Dynamics* 42.7 (July 2019), pp. 1579–1585. DOI: 10.2514/1.G003999.

[19] John A. Christian and Shane Robinson. “Noniterative Horizon-Based Optical Navigation by Cholesky Factorization”. In: *Journal of Guidance, Control, and Dynamics* 39 (Aug. 2016), pp. 1–9. DOI: 10.2514/1.G000539.

[20] William M. Jr. Owen. “Methods of Optical Navigation”. In: *AAS Spaceflight Mechanics Conference, New Orleans, Louisiana.* 2011.

[21] Yi Ma, Stefano Soatto, and Jana Kośecká. *An Invitation to 3D Vision: From Images to Geometric Models.* Springer, 2004. ISBN: 9781441918468.

[22] John A. Christian. “Optical Navigation Using Planet’s Centroid and Apparent Diameter in Image”. In: *Journal of Guidance, Control, and Dynamics* 38 (2015), pp. 192–204. DOI: 10.2514/1.G000872.

[23] Andrew Fitzgibbon, Maurizio Pilu, and Robert B. Fisher. “Direct Least Square Fitting of Ellipses”. In: 21.5 (1999). DOI: 10.1109/34.765658.

[24] Radim Halir and Jan Flusser. *Numerically Stable Direct Least Squares Fitting Of Ellipses.* 1998.

[25] Francois Chollet. *Deep Learning with Python.* Manning, 2018. ISBN: 9781617294433.

[26] A. M. Turing. “Computing Machinery and Intelligence”. In: *Mind 59* 236 (1950), pp. 433–460.

[27] Tue Herlau, Mikkel N. Schmidt, and Morten Mørup. *Introduction to Machine Learning and Data Mining Lecture notes.* 2020.

[28] Gareth James et al. *An Introduction to Statistical Learning.* Springer, 2014. ISBN: 9781461471370.

[29] Charu C. Aggarwal. *Neural Networks and Deep Learning.* Springer, 2018. ISBN: 9783319944623.

[30] Ian Goodfellow et al. “Maxout Networks”. In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. 3. Atlanta, Georgia, USA: PMLR, June 2013, pp. 1319–1327. URL: `https://proceedings.mlr.press/v28/goodfellow13.html`.

[31] David H. Hubel and Torsten N. Wiesel. “Receptive Fields of Single Neurons in the Cat’s Striate Cortex”. In: *Journal of Physiology* 148 (1959), pp. 574–591.

[32] Y. Lecun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: `10.1109/5.726791`.

[33] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012.

[34] Aurélien Géron. *Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow*. O’Reilly, 2019. ISBN: 9781492032649.

[35] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: `10.1109/CVPR.2016.90`.

[36] Thibaud F. Theil. “Optical Navigation using Near Celestial Bodies for Spacecraft Autonomy”. PhD thesis. ISAE-Supaero, 2020.

[37] Daniele Mortari et al. “Image Processing of Illuminated Ellipsoid”. In: *Journal of Spacecraft and Rockets* 53.3 (June 2016), pp. 448–456. DOI: `10.2514/1.A33342`.

[38] N. Otsu. “A Threshold Selection Method from Gray-Level Histograms”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 9.1 (1979), pp. 62–66. DOI: `10.1109/TSMC.1979.4310076`.