SAPIENZA
UNIVERSITÀ DI ROMA

Satellite On–Board Solutions for Precise Orbit Determination on Earth and Moon Orbit

Sapienza University of Rome
PhD in Automatic Control, Bioengineering and Operations Research (XXXVI cycle)

Andrea Tantucci
ID number 1694755

Advisors
Prof. Antonio Pietrabissa
Prof. Francesco Delli Priscoli

Academic Year 2023/2024

Satellite On–Board Solutions for Precise Orbit Determination on Earth and Moon Orbit
Sapienza University of Rome
ISBN: 000000000-0

This thesis has been typeset by LaTeX and the Sapthesis class.

Version: February 3, 2024

Author's email: tantucci@diag.uniroma1.it

To my parents

Abstract

Precise Orbit Determination, which is the problem of finding the satellite ephemeris by estimating the satellite position and velocity based on Earth observations data, has always been one of the most important aspects of satellite navigation. Satellite positioning is used daily by smartphones to provide several features and also by military personnel. Both of these users require different levels of accuracy. In the last decades the increasing interest in space exploration has brought forward the necessity to provide the satellites with on–board estimation algorithms. The ability to self–estimate their position and velocity is of utmost importance in scenarios in which data from Earth are not available, like in Moon or Mars orbiting. Artificial Intelligence has proven to be one of the best solutions, able to provide the satellite with non–standard measurements. The abstraction and generalization capability of neural networks allow to perform complex tasks while satisfying real–time constraints. In this context Crater Matching is one of the most promising solutions for orbit determination.

In this thesis two different approaches for on–board Precise Orbit Determination will be proposed: one making use of standard GNSS measurements coming from Earth and the other one making use of non–standard ones provided by neural networks. In the former case an end–to–end analysis, going from satellite propagation to satellite visibility has been performed. In the latter case a first step toward the development of a full Terrain Relative Navigation system has been carried out: a benchmarking of different neural networks architectures has been performed, by using a space–qualified processor, in order to identify the best on–board solution to deal with the crater detection problem.

Two additional research projects tackling important aspects of the space domain, satellite communication and Earth observation respectively, will also be detailed. A mixed Artificial Intelligence and Reinforcement Learning solution has been proposed for the first topic with extensive simulations and comparisons to validate the approach. A full Artificial Intelligence approach has instead been taken to tackle the second project, in which a Convolutional Neural Network has been trained to detect wildfires in real–time and has been tested over a space–qualified processor to verify its feasibility to be deployed on–board.

Acknowledgments

Here i would like to thank all the people which have made possible for me to reach this goal which is the most important one in my student career.

First of all i would like to thank my professors, in particular Prof. Francesco Delli Priscoli and my supervisor Prof. Antonio Pietrabissa, which have been supporting me from the bachelor degree until now. They have always encouraged me to go forward and to give my best in everything i had to do.

I would like to thank also all the colleagues and friends i met during my study careers. In particular i would like to mention Andrea with whom i have shared all these years of study. He has been a source of inspiration and a good friend to rely on.

I would also like to mention all the people from Thales Alenia Space who helped me to develop the research projects presented in this thesis: Edoardo, Federica, Oreste, Tommaso, Mattia, Cristian, Salvatore and many more. They have supported me in these three years and given me the opportunity to know how does it feel to work and research in such a big company. It has been a really good learning experience.

Last but not the least i would like to thank my parents, which were the ones who made all of this possible. They were always there in the good and bad moments of my life and have always been supporting my decisions.
To them i dedicate this important step in my life and all the future ones.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

ADCS Attitude Determination and Control System.

AI Artificial Intelligence.

CCSDS Consultative Committee for Space Data Systems.

CNN Convolutional Neural Network.

COTS Commercial Off–The–Shelf.

CPU Central Processing Unit.

CV Computer Vision.

CVAI Computer Vision and Artificial Intelligence.

DoD Department of Defence.

DOP Diluition Of Precision.

ECEF Earth Centered Earth Fixed.

ECI Earth Centered Inertial.

EO Earth Observation.

ESA European Space Agency.

FPGA Field Programmable Gate Array.

FWHM Full Width at Half Maximum.

GCRF Geocentric Celestial Reference Frame.

GDOP Geometric Diluition of Precision.

GEO Geostationary Earth Orbit.

GNSS Global Navigation Satellite System.

GPS Global Positioning System.

GPU Graphics Processing Unit.

ISP Image Signal Processing.

LEO Low Earth Orbit.

LSTM Lon Short Term Memory.

LTDN Local Time at the Descending Node.

LVDS Low Voltage Differential Signaling.

MEO Medium Earth Orbit.

NASA National Aeronautic and Space Administration.

NCS Neural Compute Stick.

NN Neural Network.

ODTS Orbit Determination and Time Synchronization.

PAN Panchromatic.

PCIe Peripheral Component Interconnect Express.

PDOP Position (3D) Diluition of Precision.

POD Precise Orbit Determination.

PRN PseudoRandom Noise.

RAAN Right Ascension of the Ascending Node.

SAR Synthetic Aperture Radar.

SC SpaceCraft.

SDK Software Development Toolkits.

SNR Signal to Noise Ratio.

SSO Sun Synchronous Orbit.

TASI Thales Alenia Space Italy.

TDOP Time Diluition of Precision.

TRAN Terrain–Relative Absolute Navigation.

TRN Terrain–Relative Navigation.

TRRN Terrain–Relative Realative Navigation.

UERE User Equivalent Range Error.

UERRE User Equivalent Range Rate Error.

VPU Visual Processing Unit.

WGS World Geodetic System.

Chapter 1

Introduction

Orbit Determination (OD) is the problem of finding the satellite ephemeris by estimating the satellite position and velocity based on observations data (Figure 1.1). These observations are fed into the orbit determination algorithm and can be of different natures: ground–based observations such as azimuth, elevation, range and/or range–rate values or sensor–based observation such as altitude, orientation of the body and angular rate of the satellite, distance from other “bodies” (Moon or Sun). The former are provided by Earth ground tracking stations while the latter are provided by the satellite on–board sensors like altimeters, Inertial Measurement Unit (IMU) devices and star trackers.

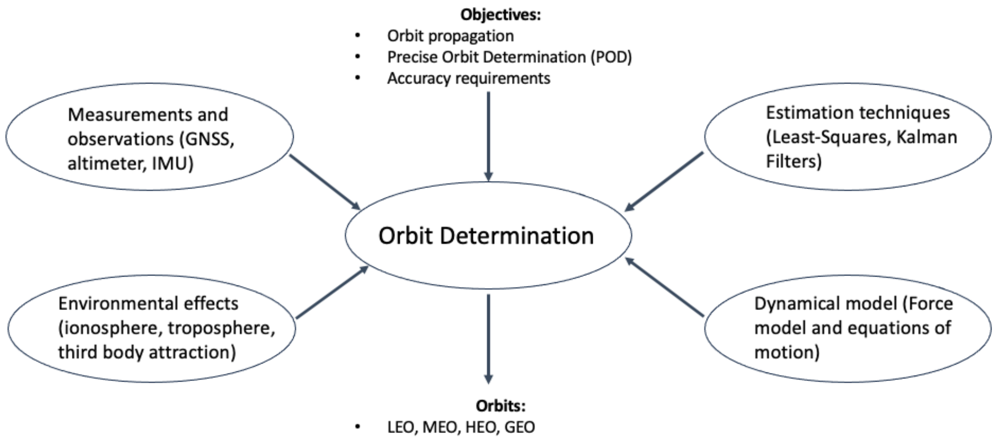

Objectives:
• Orbit propagation
• Precise Orbit Determination (POD)
• Accuracy requirements
Measurements and observations (GNSS, altimeter, IMU)
Estimation techniques (Least-Squares, Kalman Filters)
Orbit Determination
Environmental effects (ionosphere, troposphere, third body attraction)
Dynamical model (Force model and equations of motion)
Orbits:
• LEO, MEO, HEO, GEO

Figure 1.1. Orbit Determination Problem and Topics.

The increase in exploration and scientific space missions has brought up the need to develop algorithms which are able to estimate the satellite pose with high precision, in particular for LEO satellites. These are named Precise Orbit Determination (POD) algorithms. They make use of nonlinear estimation algorithms to estimate the state of the satellite non–linear propagation model. Until recent times, most of the processing was performed on ground due to the low computational power of satellite on–board processors.

The improvement of on–board technology has opened up another possibility in satellite navigation that is, trying to execute the processing which was performed on ground, on–board the satellite. This problem is named the Orbit Navigation (ON) problem. It relies on the direct on–board processing of avionic sensors and auxiliary observables to obtain real–time high–accuracy satellite positioning. The primary aim is to support platform autonomous operations as trajectory and attitude control tasks, even though payload and data handling can benefit of a precise position tag. This is very important for deep space exploration missions in which the ability of the satellite of auto–determine its position is of utmost importance.

Modern OD systems, which are the result of methodologies developed over the past 50 years [1], have demonstrated, in the framework of many research programs carried out by DoD, NASA and ESA, the advantage of a generalized Earth orbit determination scheme dealing with a large variety of space missions and orbital regimes (LEO, MEO, GEO, etc.). Their propagation and estimation open architectures allow suitable orbit models to be selected and target state vector rearrangement: it is possible in this way to properly calibrate the model with respect to the orbit regime peculiarities by manipulating a list of operating parameters. The capability of handling different kinds as well as different combinations of observables completes the generalized OD paradigm: measurement reconstruction patterns modelling allows the optimization of the performance with respect to both random and systematic errors.

The possibility to have an OD–based scheme is becoming of high interest also for on–board ON systems. With the recent interest of space companies in the Moon and Mars exploration and exploitation, the need to find auxiliary observables in order to improve the satellite pose estimation has become an important aspect for the development of new navigation algorithms. In this framework the use of machine learning approaches, in particular NNs and CNNs, is of great interest since they are able to perform on–board computations satisfying the real–time constraints imposed by ON systems thanks to the use of ad–hoc processing units (GPUs or VPUs) mounted on the on–board processors. The use of NN for on–board tasks allows to obtain non–standard measurements, like for example Moon or Mars craters' position by performing image segmentation or object detection on the Moon's surface. This feature can be used to boost the ON algorithms by improving the position and velocity estimation of the satellite which, in case of a lack of Earth measurement data, would be performed only by means of the on–board sensors like altimeters or star trackers.

In this work two novel approaches related to the POD problem are proposed:

- the first is an on–board POD algorithm for a LEO satellite based on multiple GNSS observables. In this project an ODTS algorithm based on multiple GNSS measurements will be presented. An Extended Kalman Filter (EKF) will be used for state estimation. Moreover a detailed mathematical description of the POD algorithm, i.e. satellite equations of motions, visibility analysis and EKF formulation, will be given along with some simulations on the filter accuracy and its robustness in the presence of error contributions in the measurement equations;
- the second is an on–board POD algorithm in a Moon–based scenario which relies on auxiliary observables obtained by exploiting AI algorithms. In this thesis a first step towards a full TRN pipeline is presented, in particular the crater detection processing block of the TRN pipeline has been analyzed. Several neural networks have been tested, spanning from image segmentation to object detection NNs and their performances in terms of computational complexity have been compared. The results are shown in this thesis along with a discussion on their capability to handle on–board constraints. All the tests have been performed over a space–qualified processor.

In addition two more research projects will be presented:

- the first one deals with the problem of free–space optical satellite communications. The proposed approach makes use of artificial intelligence and reinforcement learning in order to develop an optimal data–path control law which allows the correct exchange of signals between two distant areas of the globe. The communication link is established by means of LEO satellite constellations. Several case studies will be presented and the solution will be compared with other standard benchmark approaches;
- the second one focuses on the Earth observation domain. The solution was developed as an answer to an ESA challenge looking for innovative AI–based on–board applications for the Phisat–2 mission. The challenge is called *OrbitalAI*. The idea was to develop a neural network able to identify the presence of wildfires in an area. This application will be useful to the relevent authorities

in order to act in time and prevent further disasters in the nearby areas. A detailed description of the training and validation process of the chosen model is presented. This solution has also been tested over a space–qualified processor.

The results presented in Chapters 3 and 5 have been developed in collaboration with Thales Alenia Space which allowed to perform tests on space–qualified hardware; the work detailed in Chapter 2 has resulted in a publication at the Mediterranean Control Conference 2023, while the research detailed in Chapter 4 has been submitted to the International Journal of Satellite Communications and Networking.

The thesis is organized in the following way: in Part I the two on–board solutions for satellite POD problem are presented; Part II instead details the two additional research projects presented above; in Chapter 6 the final summary and considerations related to this thesis are presented.

Part I

On–Board Precise Orbit Determination on Earth and Moon Orbit

Chapter 2

Precise Orbit Determination on LEO Satellite using Pseudorange and Pseudorange-Rate Measurements

2.1 State of the art

The POD problem is one of the most important aspects characterizing satellite mission operations. It consists in the accurate prediction of the satellite ephemeris by estimating the satellite position and velocity based on a sequence of GNSS observations.

GPS receivers are widely used on LEO satellites due to the global coverage offered by the GPS constellation which allows a wide number of observations to be processed at the same time, differently from the MEO and GEO orbits, in which a reduced number of usable satellites, high values of GDOP and significant GNSS outage periods do not allow a precise estimation of the satellite position and velocity.

A 10 m and 0.1 m/s navigation accuracy for position and velocity, respectively, is generally considered adequate for Attitude and Orbit Control Systems (AOCS). Following the deactivation of selective availability, this accuracy can readily be provided by the kinematic navigation solution using a single–frequency GPS receiver. However, much higher accuracy is required in many on–board navigation tasks, such as SAR interferometry or atmospheric sounding. These cases call for a sub–decimeter position accuracy and a sub–mm/s velocity knowledge. In the past, this accuracy could be obtained in a ground–based reduced dynamic orbit determination using dual–frequency carrier phase measurements along with precise GPS ephemeris products and auxiliary environmental information. To improve the accuracy of the on–board navigation solutions, Kalman filtering techniques have been broadly used, finding applications in both attitude [2, 3] and orbit propagation control schemes [4, 5]. In [6] the author compares the use of the Unscented Kalman Filter (UKF) approach with the EKF one with different types of measurements: GPS navigation solutions (position and velocity of the satellite obtained by least squares of the GNSS measurements) and pseudorange measurements. It is shown that the latter method improves the accuracy: in particular, a maximum accuracy of about 12 m for the position and 0.0159 m/s for the velocity at convergence is obtained. [7] proposes a novel approach using the Schmidt Consider Kalman Filter (CKF). This approach is able to obtain a high level of accuracy also in the presence of an high number of uncertainties in the equations of motion, which need to be estimated by the Kalman Filter, by partitioning the filter state into actual estimation variables and uncertain parameters. The latter ones are maintained constant during the propagation and are not corrected by the measurements.

2.2 Original Contribution

In this work, an EKF estimation algorithm has been considered. The measurements which have been used are the pseudorange and the pseudorange–rate. On the one hand, it will be shown that the use of multiple GNSS measurements increases the accuracy by a huge margin with respect to using only one of them as in [6], which uses the pseudorange measurements only. On the other hand, [8, 9] use the carrier–phase instead of the pseudorange–rate since it allows to mitigate some of the measurement errors. However, the use of carrier–phase measures introduces new state variables to be estimated in the form of ambiguity biases, whose number varies at every observation instant since it corresponds to the number of tracked satellites. The dynamic augmentation of the filter state increases the storage and computational load of the satellite hardware, which might be significant in small LEO satellites. The use of the pseudorange–rate instead, allows to obtain an accuracy below the 1 m threshold for the position and below the 0.001 m/s threshold for the velocity, without increasing the filter state dimension as in the previous approaches, therefore reducing the storage and computational load for the on–board processor. This is also an improvement with respect to [5], where an accuracy of 10 meters (1 σ) for the position and of 0.01 m/s (1 σ) threshold for the velocity has been obtained by using, as measurements, the pseudorange and the Doppler.

Two different sets of simulations have been performed in order to verify the robustness of the filter to the presence of different error contributions in the measurements:

- in the first case the error contributions correspond to the receiver clock bias and drift errors Δt_R and $\Delta \dot{t}_R$, the receiver measurement errors ϵ_ρ and $\epsilon_{\dot{\rho}}$, the ionospheric and multipath errors;
- in the second case, in addition to the previous contributions, the receiver relativistic error and the Shapiro effect have been added.

Since the focus of this research is to validate the use of the GNSS measurements, the presence of uncertainties in the equations of motion has been limited in order to keep the filter state at a reasonable dimension. Nevertheless, if one wants to introduce additional uncertainties, a more robust filter might be used, as for example the CKF.

This work has been published in the proceedings of the Mediterranean Control Conference (MED) 2023 [10].

2.3 Implementation

In this section a detailed description of the implementation of the ODTS algorithm will be presented. First the satellite equations of motion will be detailed followed by a dissertation over the EKF formulation which has been used as state estimator.

2.3.1 Mathematical Model of the Satellite Motion

The equation of motion of a satellite has the following closed form:

$$\dot{\mathbf{x}} = f(\mathbf{x}, t) \tag{2.1}$$

with $\mathbf{x} = (\mathbf{r},\, \mathbf{v})$ and $\dot{\mathbf{x}} = (\dot{\mathbf{r}},\, \dot{\mathbf{v}}) = (\mathbf{v},\, \mathbf{a})$ being the state and its derivative respectively. In particular $\mathbf{r}$, $\mathbf{v}$ and $\mathbf{a}$ represent the position, velocity and acceleration of the satellite.

Newton's law of motion states that the force on a body equals the product of that body's mass multiplied by its acceleration. When the position is measured in an ECI reference frame, such as J2000 or GCRF, this law can be written as follows:

$$m\ddot{\mathbf{r}} = m\frac{d^2\mathbf{r}}{dt} = \mathbf{F} \tag{2.2}$$

where m is the mass of the body, $\mathbf{F}$ is the total force acting on the body and $\mathbf{r}$, $\ddot{\mathbf{r}}$ are the position and the acceleration vectors of the satellite. Both $\mathbf{r}$ and $\mathbf{F}$ are expressed in ECI. In the case of an Earth orbiting satellite, there are several contributors to the total force **F**. These include the gravitational attraction of the Earth, Sun, Moon and other celestial bodies, the atmospheric drag and solar radiation pressure and any thrust produced by the satellite while performing maneuvers or adjusting its attitude. Assuming that only the Earth's gravitational attraction acts on the satellite and that the Earth is spherically symmetric, Newton's law of gravity gives the force between Earth and satellite:

$$\mathbf{F} = -\frac{GMm}{\|\mathbf{r}\|^3}\mathbf{r} \tag{2.3}$$

where M is the mass of the Earth, G is the gravitational constant, $\mathbf{r}$ is the position of the satellite in ECI and $\|\mathbf{r}\|$ is the distance of the satellite from the center of the Earth. Defining the Earth's gravitational parameter as $\mu = GM$ and using Newton's law of motion, the acceleration of the satellite is:

$$\ddot{\mathbf{r}} = -\mu\frac{\mathbf{r}}{\|\mathbf{r}\|^3}. \tag{2.4}$$

This acceleration would result in Keplerian orbital motion, which has an easy analytical solution. If perturbations are considered, (2.4) becomes:

$$\ddot{\mathbf{r}} = -\mu\frac{\mathbf{r}}{\|\mathbf{r}\|^3} + \mathbf{P}(\mathbf{r}, \mathbf{v}, \Delta C_\mathrm{d}, \Delta C_\mathrm{sp}, t) \tag{2.5}$$

where $\mathbf{P}(\mathbf{r}, \mathbf{v}, \Delta C_\mathrm{d}, \Delta C_\mathrm{sp}, t)$ is the total perturbing acceleration due to all effects other than the spherical mass distribution of the Earth. ΔC_d and ΔC_sp are the drag and solar pressure coefficients respectively and t is the time. Because of the presence of these additional accelerations, the resulting orbital motion deviates from Keplerian orbits. The solution of the equation of motion is not analytical anymore but, provided that the total perturbing accelerations are known exactly, this equation provides an exact description of the satellite motion.

The perturbations modeled in this work are the following ones:

1. $\mathbf{a}_{\text{GRAV-NS}}(\mathbf{r}, t) \rightarrow$ *Non–spherical Earth gravitational potential.* The Earth gravitational potential can be written as a series of harmonics in order to take into account the actual mass distribution of the Earth. This term depends only on the ECEF position of the satellite (so it depends on the ECI position and on time for the ECI to ECEF transformation). The EGM2008 model is used [11];

2. $\mathbf{a}_{\text{DRAG}}(\mathbf{r}, \mathbf{v}, \Delta C_\mathrm{d}, t) \rightarrow$ *Atmospheric drag* on the satellite. The modified Harris–Priester model is used for evaluation of the atmosphere density [12]. Drag

depends on the satellite position (for calculating the atmospheric density), on the satellite velocity (for calculating the relative velocity between the satellite and atmosphere) and on time (for calculating the position of the Sun which is needed for evaluating the atmospheric density). The drag coefficient ΔC_d is estimated by the EKF in order to account for all the uncertainties linked to the drag coefficient itself and drag equivalent surface;

3. $\mathbf{a}_\text{TB}(\mathbf{r}, t) \rightarrow$ *Third body attraction.* The gravitational attraction of the Sun and the Moon are considered. Attraction from other celestial bodies (like Jupiter) are neglected since they have no effect on the POD solution. The perturbation depends only on the satellite position and time (for Sun and Moon position estimation);

4. $\mathbf{a}_\text{SP}(\mathbf{r}, \Delta C_\text{sp}, t) \rightarrow$ *Solar radiation pressure.* It depends only on satellite position and time (for Sun position estimation). The solar pressure coefficient ΔC_sp is estimated by the EKF in order to account for all the uncertainties linked to the thermo–optical properties of the surfaces and the solar pressure equivalent surface.

More details on the modeling of these perturbations are presented in Appendix A.

The total perturbing acceleration is then:

$$\begin{gathered}\mathbf{P}(\mathbf{r}, \mathbf{v}, \Delta C_\text{d}, \Delta C_\text{sp}, t) = \mathbf{a}_\text{GRAV-NS}(\mathbf{r}, t) + \mathbf{a}_\text{DRAG}(\mathbf{r}, \mathbf{v}, \Delta C_\text{d}, t) + \\ \mathbf{a}_\text{TB}(\mathbf{r}, t) + \mathbf{a}_\text{SP}(\mathbf{r}, \Delta C_\text{sp}, t)\end{gathered}. \tag{2.6}$$

In the following analysis the spherical and non–spherical parts of the Earth gravitational acceleration are calculated together by the same algorithm:

$$\mathbf{a}_\text{GRAV}(\mathbf{r}, t) = -\mu \frac{\mathbf{r}}{\|\mathbf{r}\|^3} + \mathbf{a}_\text{GRAV-NS}(\mathbf{r}, t). \tag{2.7}$$

So (2.5) can be rewritten as:

$$\ddot{\mathbf{r}} = \mathbf{a}_\text{GRAV}(\mathbf{r}, t) + \mathbf{a}_\text{DRAG}(\mathbf{r}, \mathbf{v}, \Delta C_\text{d}, t) + \mathbf{a}_\text{TB}(\mathbf{r}, t) + \mathbf{a}_\text{SP}(\mathbf{r}, \Delta C_\text{sp}, t). \tag{2.8}$$

2.3.2 On–Board Navigation Algorithm

In this section an overview of the EKF formulation will be presented along with its implementation: the state of the filter will be defined along with the measurement equations used as inputs to the filter itself.

Extended Kalman Filter Implementation

Necessarily, due to the large nonlinearities in the GNSS observation equations and in the dynamical model of the satellite motion, a nonlinear filter has to be adopted. The EKF is a dynamical filter which is sub–optimal, since it relies on the linearization applied to the nonlinear discrete filter dynamics [13–15]:

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{w}_k \tag{2.9}$$

$$\mathbf{y}_k = h(\mathbf{x}_k) + \mathbf{v}_k \tag{2.10}$$

where f and h are the differentiable state transition and observation functions, respectively, $\mathbf{w}_k$ and $\mathbf{v}_k$ are process and measurement noises at time k which are assumed to be zero–mean and gaussian with covariances Q_k and R_k respectively, and $\mathbf{x}_k$, $\mathbf{y}_k$ and $\mathbf{u}_k$ are the state, output and control vectors of the filter at time k.

The EKF algorithm consists of two main steps: prediction and correction.

Prediction Step

The prediction step consists of the propagation of the nonlinear dynamics and of the state–error covariance matrix, computed by the following discrete equations:

$$\hat{\mathbf{x}}_f(k+1) = f(\hat{\mathbf{x}}_f(k), \mathbf{u}(k)) \tag{2.11}$$

$$P(k+1) = \Phi P(k) \Phi^T + Q \tag{2.12}$$

given some initial conditions $\hat{\mathbf{x}}_f(0)$, $P(0)$. $\hat{\mathbf{x}}_f$ is the estimated filter state, P is the state–error covariance matrix, Φ is the state transition matrix and Q is the state–noise covariance matrix. In practice, both P and Q are usually considered as diagonal or block diagonal matrices. The propagation of the filter dynamics (2.11) is performed by numerical integration of the equations of motion (2.8). The integration is performed by using a Runge–Kutta method of order four. The state transition matrix in Equation (2.12) is obtained by numerically integrating the following differential equation:

$$\frac{d}{dt}\Phi(t, t_0) = \frac{\partial f(\hat{\mathbf{x}}_k, \mathbf{u}_k)}{\partial \hat{\mathbf{x}}_{k-1}} \Phi(t, t_0) \tag{2.13}$$

with initial condition $\Phi(t_0, t_0) = I$, where I denotes the identity matrix of appropriate dimensions.

Correction Step

The correction step is performed when a set of measures with an associated Time Tag (time at which the measurements have been taken) are sent to the filter by the receiver, and consists of the following equations:

$$\mathbf{z} = \mathbf{y} - \mathbf{h}(\hat{\mathbf{x}}_k) \tag{2.14}$$

$$K = PH^T(R + HPH^T)^{-1} \tag{2.15}$$

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + K\mathbf{z} \tag{2.16}$$

$$P^+ = (I - KH)P^- \tag{2.17}$$

where $\mathbf{z}$ is the measurement residual, $\mathbf{y}$ is the actual measurement vector, $H = \dfrac{\partial h(\hat{\mathbf{x}}_k)}{\partial \hat{\mathbf{x}}_{k-1}}$ is the measurement jacobian w.r.t. the state, R is the measurement error covariance matrix, K is the Kalman gain, $\hat{\mathbf{x}}_k^-$ and $\hat{\mathbf{x}}_k^+$ are the states before and after the correction, respectively. Since, in a real time environment, the time of arrival of the measures is not known apriori, there is an unknown delay in the measurements which has to be taken into account. To overcome this problem the following three–steps correction has been implemented (see Figure 2.1):

1. The first step is a backward propagation of the filter state from the current time to the Time Tag of the measurements;
2. The correction step is performed on the state obtained on the previous step;
3. The corrected state is propagated forward from the measurement Time Tag to the current time.

After the correction steps have been performed the state is propagated as in the previous section until a new set of measurements is detected by the receiver. This method allows to always have a corrected estimate at the current time.

UDU Covariance Factorization

An alternative approach widely used in aerospace engineering applications is the so called UDU formulation of the Kalman filter [16–18]. The idea is to replace the covariance matrix P by two factors: a diagonal matrix D and an upper triangular matrix U with ones on the main diagonal, such that:

$$P = UDU^T. \tag{2.18}$$

Whereas the UDU factorization improves the computational stability and efficiency of large navigation filters, it was originally used in a batch formulation [19]. However,

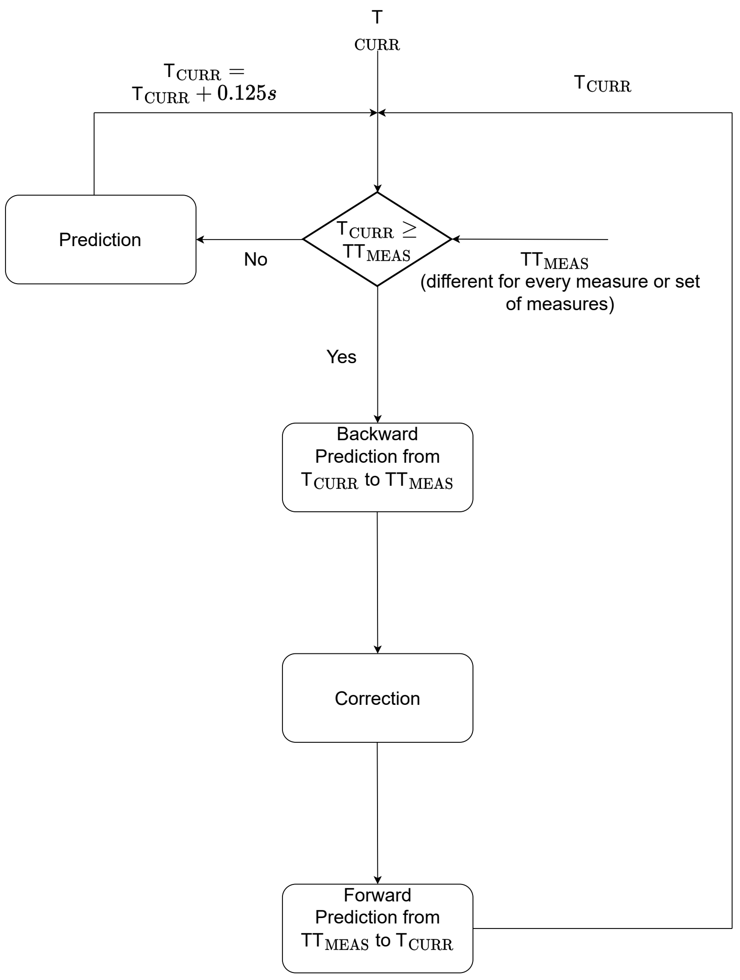

T CURR
T CURR = T CURR + 0.125s
T CURR
Prediction
No
T CURR ≥ TT MEAS
TT MEAS
(different for every measure or set of measures)
Yes
Backward Prediction from T CURR to TT MEAS
Correction
Forward Prediction from TT MEAS to T CURR

Figure 2.1. Extended Kalman Filter Implementation Flow Chart.

this formulation lent itself to sequential implementations, well–suited for platforms where both computational stability and numerical efficiency are at a premium.

The UDU formulation of the Kalman filter has great numerical stability properties [17]: it insures symmetry of the covariance by construction, and it requires a trivial check and correction to ensure semi–positive definiteness (it suffices to enforce that the diagonal elements of D remain non–negative). The UDU formulation is free from square root operations, making it computationally cheaper than other formulations like the Cholesky approach [20, 21]. For these reasons the UDU has endured as one of the preferred practical implementation of Kalman filters in aerospace applications.

The UDU approach to propagate U and D forward in time makes use of the Modified Weighted Gram–Schmidt (MWGS) orthogonalization algorithm, avoiding loss of orthogonality due to round–off errors [22]. Measurements are processed one at the time as scalars by noting that when R is diagonal the update is obtained by recursively processing one element of $\mathbf{y}$ at a time, using the corresponding row of H and diagonal element of R. The measurement residual covariance matrix $W = R + HPH^T$ thus becomes a scalar, and the quantity $PH^T = \mathbf{w}$ becomes a vector; thus each of the scalar updates takes the form:

$$P^+ = P^- - \frac{1}{W}\mathbf{w}\mathbf{w}^T. \tag{2.19}$$

Since P^- is updated with a rank one matrix $\frac{1}{W}\mathbf{w}\mathbf{w}^T$, this is called rank one update. In Appendix B a detailed description of the factorization, prediction and update algorithms is presented.

Filter State and State Propagation

In the orbit navigation scenario, there are a lot of parameters which need to be taken into account for a good estimation of the satellite position and velocity. Since GNSS measurement equations are considered in the navigation algorithm, the time synchronization biases have to be considered as well. A standard solution consists in including both the GPS clock bias Δt_{R} and clock drift $\Delta\dot{t}_{\mathrm{R}}$ in the estimation state [5, 8, 23]. The estimation of these two parameters allows a more precise replica of the pseudorange and pseudorange–rate measurements for the filter algorithm. By adding the drag and solar coefficients C_{d} and C_{sp}, respectively, the filter state consists of 10 scalar variables:

$$\mathbf{x} = [\mathbf{r}, \dot{\mathbf{r}}, C_{\mathrm{d}}, C_{\mathrm{sp}}, \Delta t_R, \Delta\dot{t}_R]^T. \tag{2.20}$$

where $\mathbf{r} = \begin{pmatrix} r_x & r_y & r_z \end{pmatrix}^T$ and $\dot{\mathbf{r}} = \begin{pmatrix} \dot{r}_x & \dot{r}_y & \dot{r}_z \end{pmatrix}^T$. The state of the filter is propagated by numerically integrating the following differential equations:

$$\frac{d\mathbf{r}}{dt} = \dot{\mathbf{r}} \tag{2.21}$$

$$\frac{d\dot{\mathbf{r}}}{dt} = \mathbf{a}_{\mathrm{GRAV}} + \mathbf{a}_{\mathrm{S}} + \mathbf{a}_{\mathrm{M}} + \mathbf{a}_{\mathrm{DRAG}} + \mathbf{a}_{\mathrm{SP}} \tag{2.22}$$

$$\frac{dC_{\mathrm{d}}}{dt} = 0 \tag{2.23}$$

$$\frac{dC_{\mathrm{sp}}}{dt} = 0 \tag{2.24}$$

$$\frac{d\Delta t_{\mathrm{R}}}{dt} = \Delta\dot{t}_{\mathrm{R}} \tag{2.25}$$

$$\frac{d\Delta\dot{t}_{\mathrm{R}}}{dt} = \dot{d} \tag{2.26}$$

where $\mathbf{a}_\mathrm{S}$ and $\mathbf{a}_\mathrm{M}$ are the third body accelerations due to Sun and Moon gravity respectively (together they form the term $\mathbf{a}_\mathrm{TB}$), $\dot{d}$ is the clock frequency aging, which is a commendable parameter and varies from receiver to receiver.

Measurement Equations

The filter measurement equations account for GNSS observed variables which are collected by the GNSS receiver, namely, the pseudorange ρ_i and the pseudorange–rate $\dot{\rho}_i$. From now on, to differentiate the GNSS quantities from the LEO satellite ones, the former will be identified by a subscript SV (Space–Vehicle). By denoting with N the number of tracked satellites, the observation vector $\mathbf{h}(\hat{\mathbf{x}}_k)$ has the following form:

$$\mathbf{h}(\hat{\mathbf{x}}_k) = [\hat{\rho}_1(\hat{\mathbf{x}}_k) \ \dots \ \hat{\rho}_N(\hat{\mathbf{x}}_k) \, \hat{\dot{\rho}}_1(\hat{\mathbf{x}}_k) \ \dots \ \hat{\dot{\rho}}_N(\hat{\mathbf{x}}_k)]^T, \tag{2.27}$$

where the nonlinear observation equations have the following form:

$$\hat{\rho}_i(\hat{\mathbf{x}}_k) = \|\mathbf{r}_\mathrm{SV} - \hat{\mathbf{r}}\| + c\hat{\Delta}t_\mathrm{R} \tag{2.28}$$

$$\hat{\dot{\rho}}_i(\hat{\mathbf{x}}_k) = \hat{\mathbf{e}}(\dot{\mathbf{r}}_\mathrm{SV} - \hat{\dot{\mathbf{r}}}) + c\hat{\Delta}\dot{t}_\mathrm{R} \tag{2.29}$$

where $\mathbf{r}_\mathrm{SV}$ and $\hat{\mathbf{r}}$ are the position vectors of the GNSS satellite and the satellite respectively, $\dot{\mathbf{r}}_\mathrm{SV}$ and $\hat{\dot{\mathbf{r}}}$ are the velocity vectors of the GNSS satellite and the satellite, respectively, $\hat{\mathbf{e}}$ is the estimated unit line–of–sight vector from the user satellite to the GNSS satellite, $\hat{\Delta}t_\mathrm{R}$ and $\hat{\Delta}\dot{t}_\mathrm{R}$ are the estimated satellite clock bias and clock drift respectively, c is the speed of light.

2.4 Simulations and Results

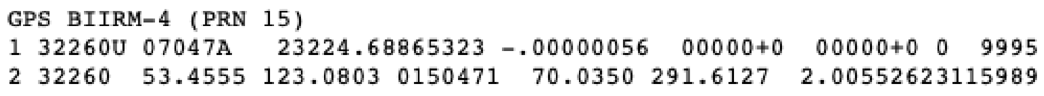
In this section the simulation environment is presented and the simulation results are shown along with a discussion on the obtained results.

2.4.1 Simulation environment

The performance assessment of the filter is performed in a simulated environment. A Matlab framework is used to simulate orbital propagation, visibility analysis and GNSS measurement equations computation. For the GNSS simulation, 30 GPS satellites have been considered.

Orbital Propagation

The orbital propagation of the satellite is performed starting from the associated Two–Line Elements (TLE) file data (see Figure 2.2).



```
GPS BIIRM-4 (PRN 15)
1 32260U 07047A   23224.68865323 -.00000056  00000+0  00000+0 0  9995
2 32260  53.4555 123.0803 0150471  70.0350 291.6127  2.00552623115989
```

Figure 2.2. Example of a Two Line Element set of data for a GPS satellite.

A two–line element set is a data format encoding a list of orbital elements of an Earth–orbiting object for a given point in time, the epoch. Using a suitable prediction formula, the state (position and velocity) at any point in the past or future can be estimated to some accuracy. TLEs can describe the trajectories only of Earth–orbiting objects. They are widely used as input for projecting the future orbital tracks of space debris for purposes of characterizing "future debris events to support risk analysis, close approach analysis, collision avoidance maneuvering" and forensic analysis [24]. A TLE set may include a title line preceding the element data, so each listing may take up three lines in the file. In particular the data are fit into 69 columns. In the following a brief description of each line is presented:

- the first line contains the satellite name;
- the second line contains data regarding the satellite launch epoch, the current epoch (year and day of the year) and other parameters like the derivatives of the mean motion;
- the third line contains the data needed to compute the orbital parameters of the satellite.

The TLE files for the GPS satellites have been obtained by [25].

A keplerian propagation has been performed to align the TLE epoch with the simulation initial epoch which is a commendable parameter: in particular only the mean motion and the RAAN have been propagated considering only the J2 effect following the approach described in [26]. Since the keplerian propagation is not very accurate, usually a TLE with an epoch as close as possible to the simulation initial epoch has to be chosen. Then the actual propagation of the LEO and GNSS satellites has been performed using ode45, which is faster and more efficient than Runge–Kutta, to integrate the equations of motion. In the GNSS case, for computational efficiency, the only perturbation considered is the gravity one; for the LEO satellite propagation all the previously defined perturbations have been taken into account into the equations of motion.

Visibility Analysis

A visibility analysis algorithm has also been implemented: it returns, at every time instant, the GNSS satellites visible from the LEO satellite, i.e. the ones sending the information needed for the estimation algorithm. In order for a signal to be sent from the GPS satellite and received from the LEO receiver, two conditions must be satisfied:

1. the GNSS–to–LEO satellite position vector must be inside both the transmitter and receiver field of view of the antennas, i.e., in their visibility cones;
2. the Carrier to Noise ratio (C/N_0) must be over a certain threshold.

To verify the first condition two additional checks have to be performed: the geometric visibility test and the electronic visibility test.

The first one ensures that the two satellites are visible from a geometric point of view, i.e. the line of sight vector connecting the two is not intersecting the Earth (see Figure 2.3). In order check this condition, the minimum distance h between the

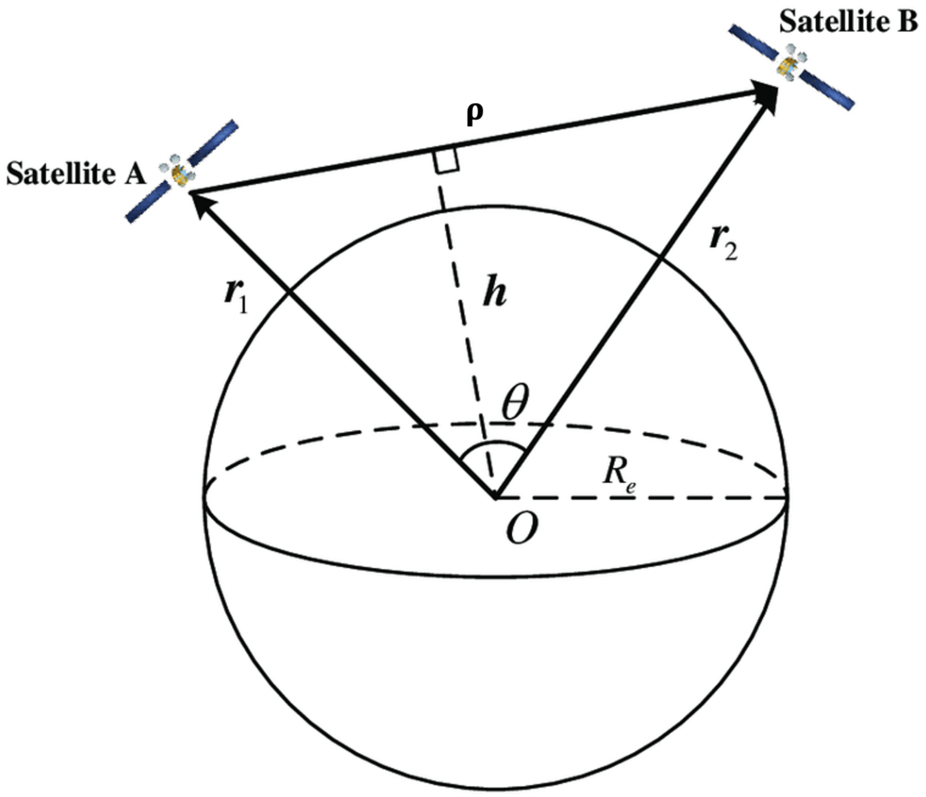


Figure 2.3. Geometric visibility between two satellites. $\mathbf{r}_1$ and $\mathbf{r}_2$ are the position vectors of the satellite with respect to the center of the Earth. $\boldsymbol{\rho}$ is the position vector between the two satellites, $\mathbf{h}$ is the minimum distance vector between $\boldsymbol{\rho}$ and the center of the Earth. R_e is the Earth radius.

line of sight vector $\boldsymbol{\rho}$ and the center of the Earth is computed. Since the Earth is a sphere, its flattening f has to be taken into account along the z coordinate of the h vector:

$$h_z = (1 + f)h_z. \tag{2.30}$$

If the norm of the minimum distance is greater than R_e, then the condition holds, otherwise the two satellites are not visible between each other.

The second check instead, ensures that the line of sight vector is inside the visibility cone of both satellites. To do this the elevation angle for both satellites must be computed and it has to be inside the range of values defined by the satellites antenna patterns (see Figure 2.4). First of all the angles α and β are computed as the

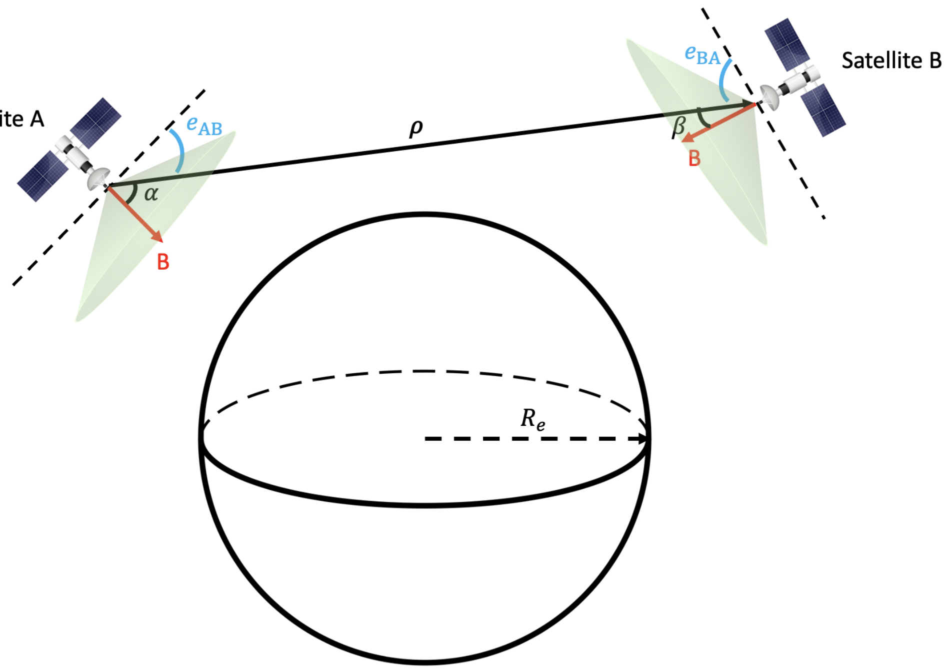


Figure 2.4. Electronic Visibility between two satellites. In red are shown the antenna boresights of the satellites, α and β are the angles between the position vector $\boldsymbol{\rho}$ and the antenna boresights, e_{AB} is the elevation angle of satellite A w.r.t. satellite B, viceversa e_{BA} is the elevation angle of satellite B w.r.t. satellite A. R_e is the Earth radius.

angles between the antenna boresights and the position vector between the satellites $\boldsymbol{\rho}$. The antenna boresights of the GNSS satellite is always pointing towards the center of the Earth, while the one of the LEO satellite is a configurable parameter. Then the elevation angles are computed as:

$$e_{\mathrm{AB}} = 90 - \alpha \tag{2.31}$$

$$e_{\mathrm{BA}} = 90 - \beta. \tag{2.32}$$

If both angles are within the respective antenna patterns, then the first condition of visibility is satisfied. In Figure 2.5 the GPS antenna gain patterns are shown, i.e. the value of the antenna gain at different elevation values. Since the patterns are symmetric, only half of the patterns are represented. GPS has two patterns: one is called Legacy and is related to a particular block of satellites (IIA) while the other one is called Improved and is related to another block of satellites (IIR) [27]. In the implementation it has been assumed that the maximum gain is at 90 degrees.

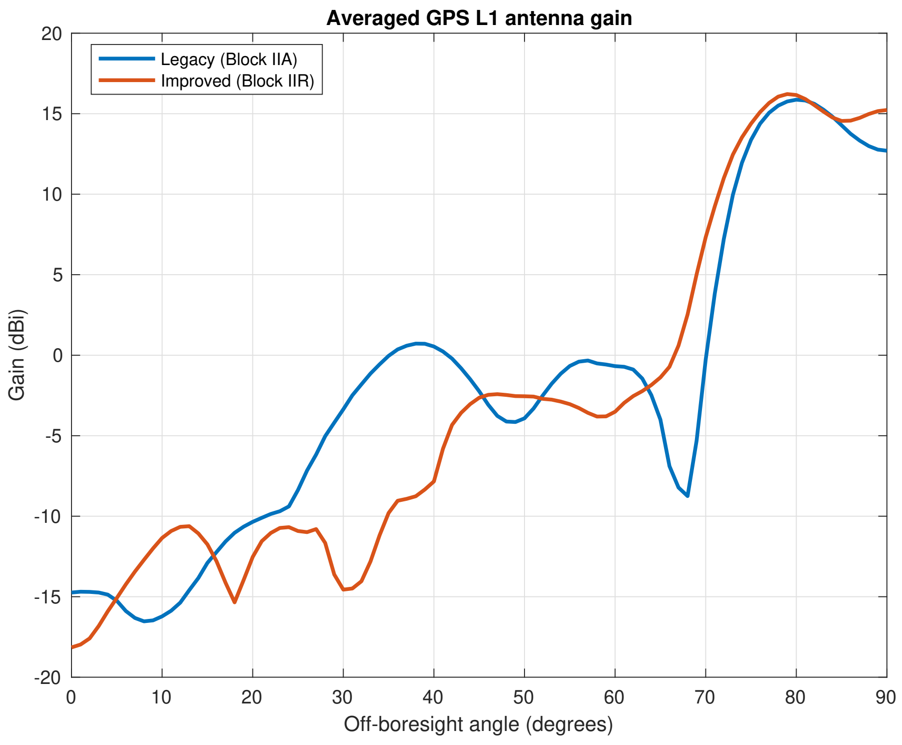


Figure 2.5. Antenna Directivity for GPS Block IIA and Block IIR.

The second condition is related to the Carrier to Noise density ratio (C/N_0). In satellite communications, the C/N_0 is the ratio of the carrier power C to the noise power density N_0, expressed in dB–Hz. It determines whether a receiver can lock on to the carrier and if the information encoded in the signal can be retrieved, given the amount of noise present in the received signal.

In particular, the C/N_0 is computed in dB–Hz using the following equation:

$$\begin{aligned} C/N_0 = P_{\mathrm{TX}} + G_{\mathrm{TX}} - L_{\mathrm{TX}} + G_{\mathrm{RX}} - L_{\mathrm{RX}} \\ - L_{\mathrm{FSL}} - N_0, \end{aligned} \tag{2.33}$$

where: P_{TX} is the transmission power of the GPS satellites expressed in dBW (dBWatt); G_{TX} and G_{RX} are the satellite and receiver antenna gains, respectively, measured in dBi (gain w.r.t. an isotropic antenna); L_{TX} and L_{RX} are the transmission and reception losses, respectively, measured in dB; N_0 is the Thermal Noise Density, expressed in Kelvin and computed as:

$$N_0 = K_{\text{B}} + T_{\text{sys}} \tag{2.34}$$

where K_{B} is the Boltzmann's constant and T_{sys} is the system noise temperature; L_{FSL} accounts for the free–space losses, and is computed in dB as:

$$L_{\text{FSL}} = 20 \log_{10}\left(\frac{4\pi d}{\lambda}\right), \tag{2.35}$$

with d being the distance between the GPS satellite and the user satellite and λ being the wavelength of the GPS L1–band (1.58 GHz).

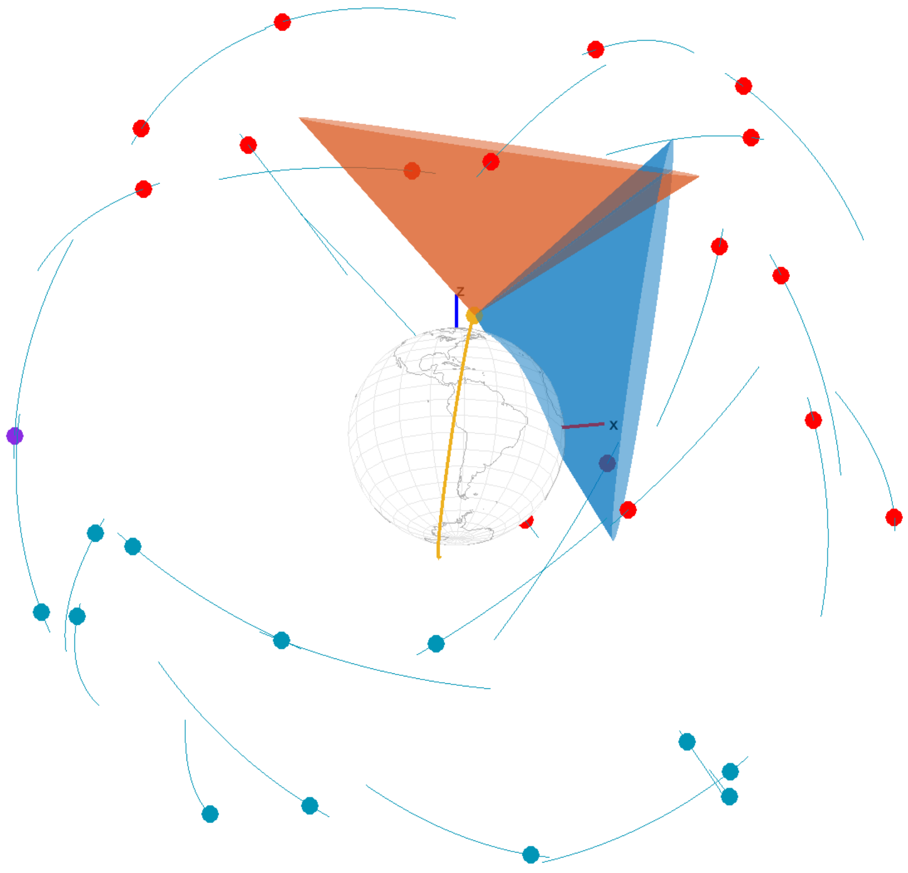


Figure 2.6. 3D–Plot of the visibility analysis in the case of multiple antennas on the LEO satellite.

In Figure 2.6 is shown a 3D–plot of the visibility analysis output in an instant of time in the case of multiple antennas mounted on the LEO satellite with different orientations. The cones are a graphical representation of their field of view. In yellow is shown the LEO satellite, in red are shown the visible satellites, i.e. the ones who have satisfied both conditions, in blue–violet are shown the satellites which have satisfied the first condition but not the second one and finally in cyan are shown the satellite which are not geometrically visible.

Geometric Diluition of Precision

In addition to the visibility algorithm and the C/N_0 computation, the GDOP has been computed. This parameter is used in satellite navigation and geomatics engineering to specify the error propagation as a mathematical effect of navigation satellite geometry on positional measurement precision. Neglecting ionospheric and tropospheric effects, the signal from navigation satellites has a fixed precision. Therefore, the relative satellite–receiver geometry plays a major role in determining the precision of estimated positions and times. Due to the relative geometry of any given satellite to a receiver, the precision in the pseudorange of the satellite translates to a corresponding component in each of the four dimensions of position measured by the receiver (i.e., x, y, z and t). The precisions of multiple satellites

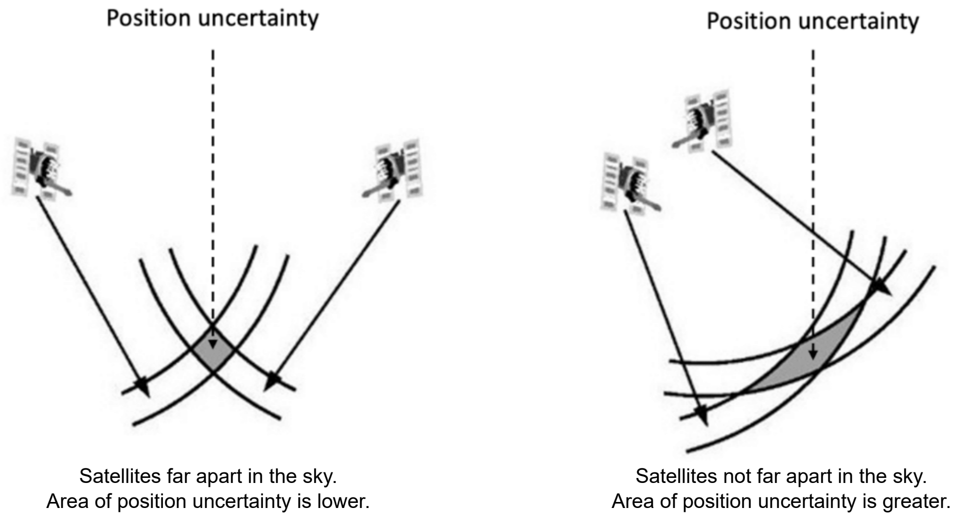

Position uncertainty
Satellites far apart in the sky.
Area of position uncertainty is lower.
Position uncertainty
Satellites not far apart in the sky.
Area of position uncertainty is greater.

Figure 2.7. Satellite geometry representations for two satellites.

in view of a receiver combine according to the relative position of the satellites to determine the level of precision in each dimension of the receiver measurement. When visible navigation satellites are close together in the sky, the geometry is said to be weak and the DOP value is high; when far apart, the geometry is strong and the DOP value is low (see Figure 2.7).

DOP value	Rating	Description
<1	Ideal	Highest possible confidence level to be used for applications demanding the highest possible precision at all times.
1–2	Eccellent	At this confidence level, positional measurements are considered accurate enough to meet all but the most sensitive applications.
2–5	Good	Represents a level that marks the minimum appropriate for making accurate decisions. Positional measurements could be used to make reliable in–route navigation suggestions to the user.
5–10	Moderate	Positional measurements could be used for calculations, but the fix quality could still be improved. A more open view of the sky is recommended.
10–20	Fair	Represents a low confidence level. Positional measurements should be discarded or used only to indicate a very rough estimate of the current location.
>20	Poor	At this level, measurements should be discarded.

Table 2.1. Geometric Diluition Of Precision possible values.

Thus a low DOP value represents a better positional precision due to the wider angular separation between the satellites used to calculate a unit's position. Other factors that can increase the effective DOP are obstructions such as nearby mountains or buildings. In Table 2.1 the possible values of GDOP are shown.

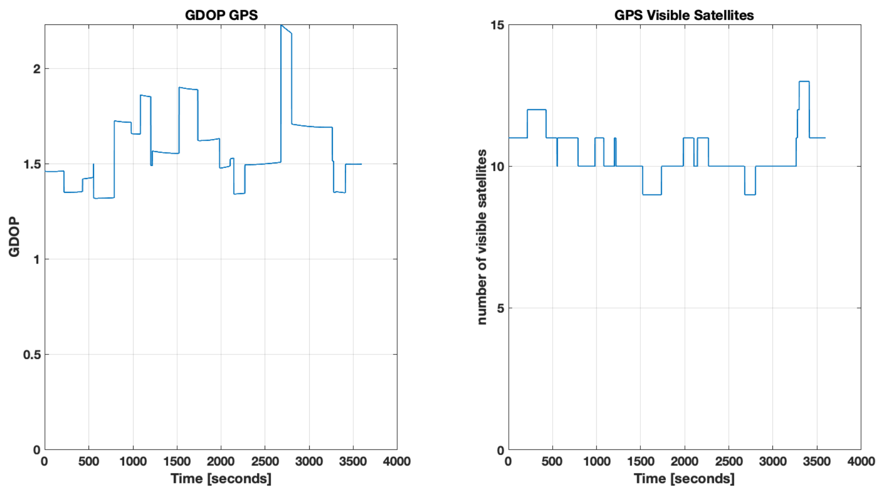


Figure 2.8. Number of visible satellites with associated GDOP value.

In Figure 2.8 the relation between number of visible satellites and GDOP can be seen. It can be observed that the GDOP is lower when the number of visible satellites is higher and is greater when the number of visible satellites is lower. Overall it can be said that the GPS constellation is well disposed in space since the GDOP values are always good.

For a detailed description of the GDOP computation see Appendix B.

Comparison with Telemetry Data

The visibility algorithm has also been tested by comparing the values obtained with real telemetry data. In particular two GPS satellites are taken as a reference: the one with PRN 13 and the one with PRN 16.

In Figures 2.9 and 2.10 are shown the visibility intervals (i.e. the intervals of time in which the receiver manages to track the satellite) of the two GPS satellites obtained by the simulated environment (in blue) and the real ones (in red). In the plots a value 1 means that the satellite is visible in an time instant, a value of 0 means that the satellite is not visible.

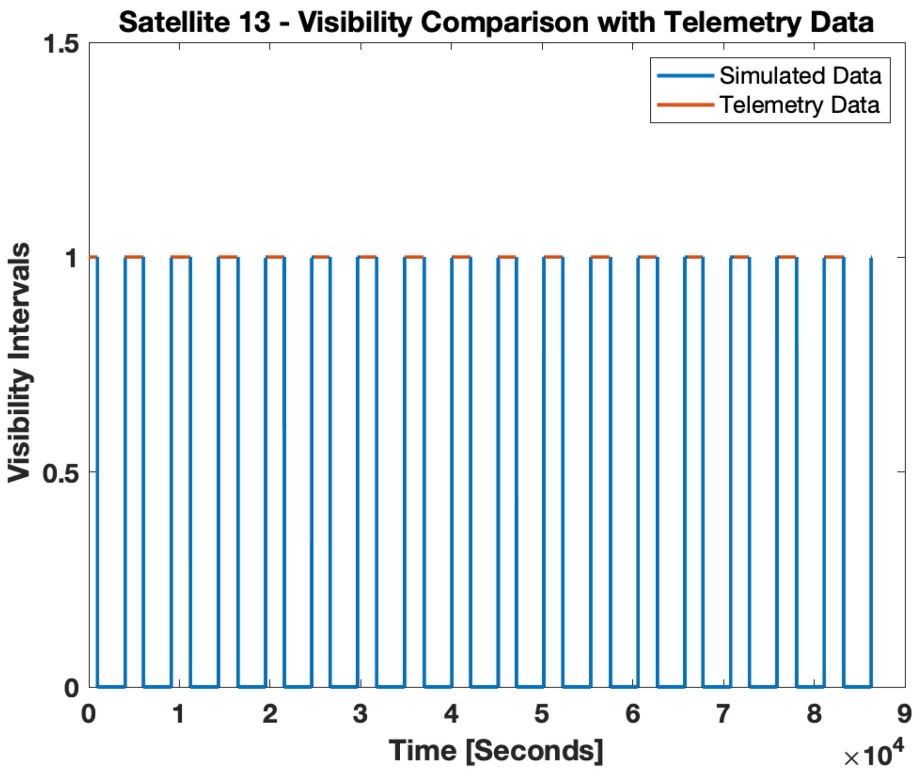


Figure 2.9. Visibility comparison for satellite 13.

It can be seen how the red intervals are contained within the blue ones. This means that, when the satellites result visible for the telemetry data, they also result visible for the simulated environment. The simulated intervals are wider with respect

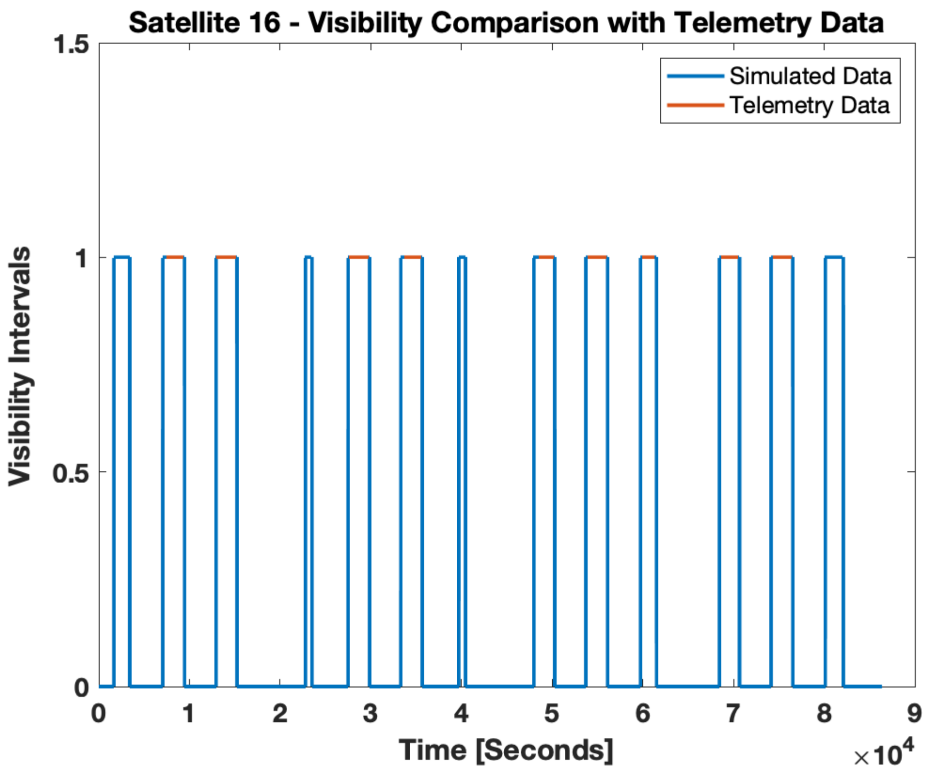


Figure 2.10. Visibility comparison for satellite 16.

to the real ones because, in a real case scenario, the receiver has a limited channel slot for the measurements, i.e. it can't receive more than a certain number of measurements at the same time. Let's call this number N_{channels}. This translates to the fact that no more than N_{channels} satellites can be tracked at the same time, while in the simulated environment this limit has not been imposed. In fact, in Figure 2.10 there are intervals of time in which the satellite is considered visible for the algorithm but not for the real data.

In Figures 2.11 and 2.12 are shown the elevation errors of the two GPS satellites with respect to telemetry data are shown. The values of the elevation error are taken in the intervals in which the satellite is visible for both the algorithm and the telemetry data. The errors are always in the interval $(-0.5, 0.5)$ degrees.

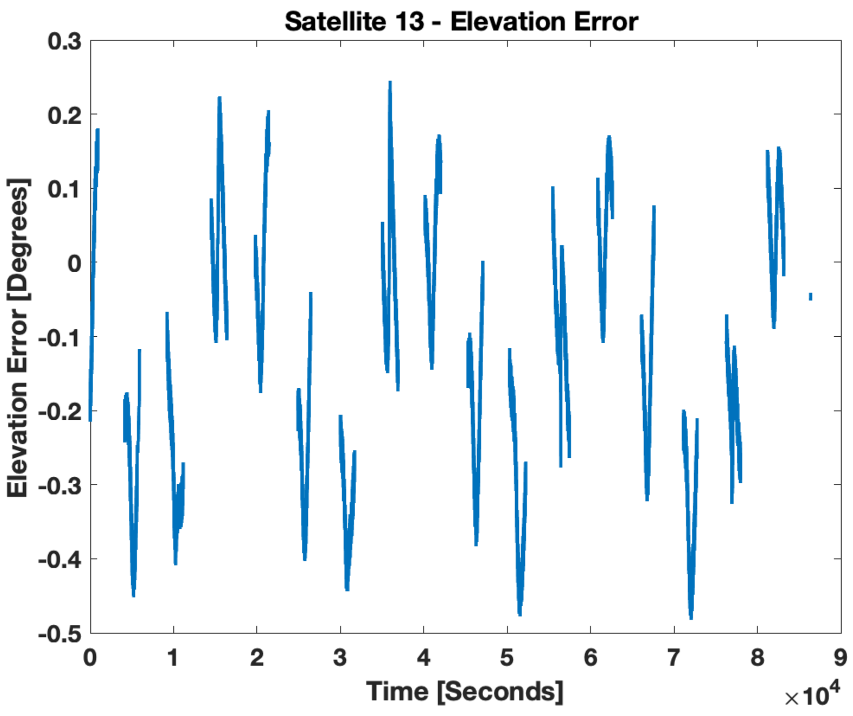


Figure 2.11. Elevation error for satellite 13.

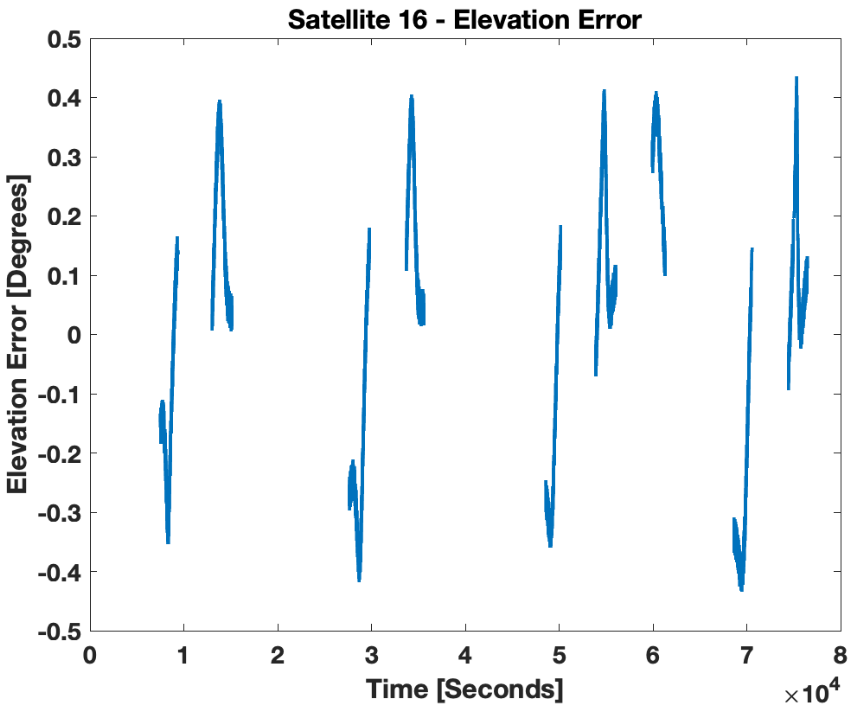


Figure 2.12. Elevation error for satellite 16.

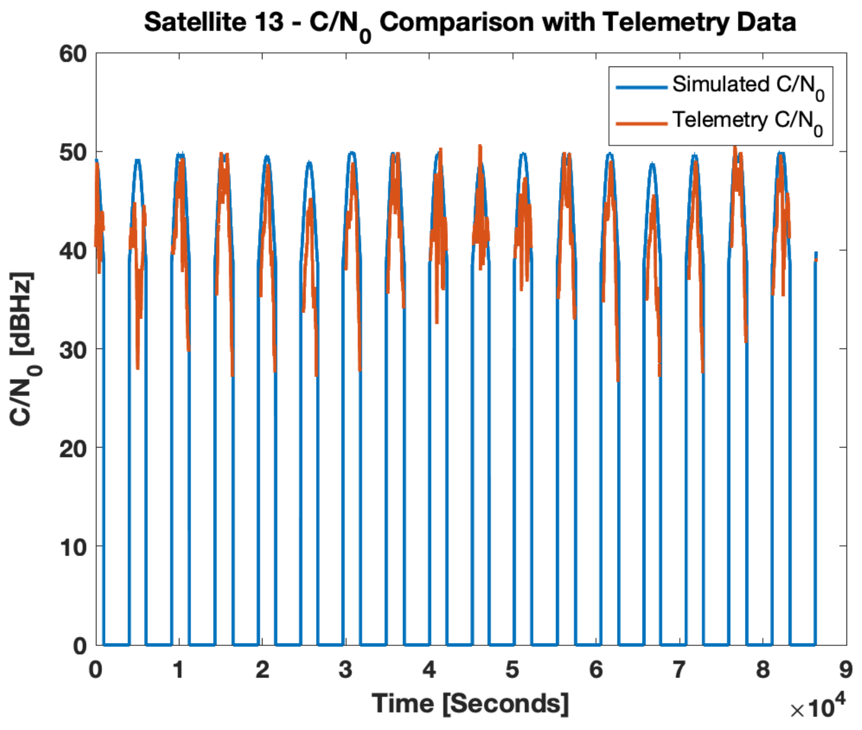


Figure 2.13. C/N_0 comparison for satellite 13.

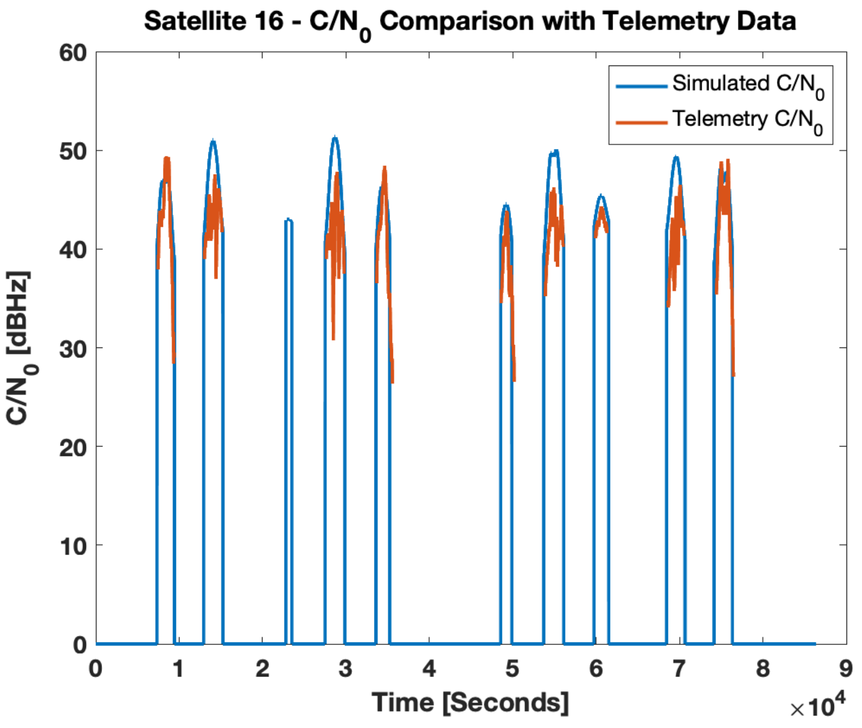


Figure 2.14. C/N_0 comparison for satellite 16.

In Figures 2.13 and 2.14 are shown the values of the C/N_0 obtained by the simulated environment with respect to the real ones obtained from telemetry data. It can be noticed that the simulated ones have a smoother behaviour. This is an expected result: in the real case the C/N_0 values are measured using electronic devices and its value oscillates in time. However it can be said that the overall behaviour of the simulated results is in line with the real case.

As a last test, the GDOP values have been compared with real data (see Figure 2.15). The presence the peaks in the error plot (the blue one) is an expected behaviour: they coincide with the instants in which the number of visible satellites changes with respect to the previous time instants. The satellite receiver has a particular procedure to handle these situations which was not implemented. So they have not to be considered as errors. In Figure 2.16 is shown a section of the previous figure in which it can be easily seen the behaviour described above. It can be seen how in the real scenario (the red one) the change in the GDOP value happens slightly after the simulated case (the orange one). This can be explained by the fact that, in the simulated case, the algorithm reacts instantly to the change in the number of visible satellites, while in the real case this in not true and there is a slight delay.

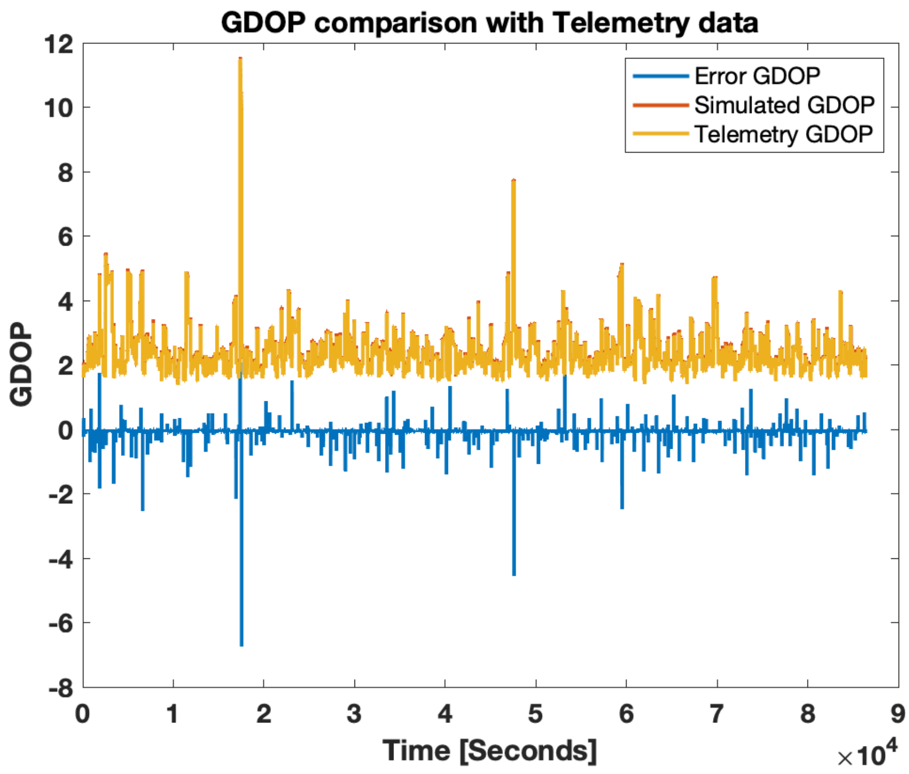


Figure 2.15. GDOP comparison with telemetry data.

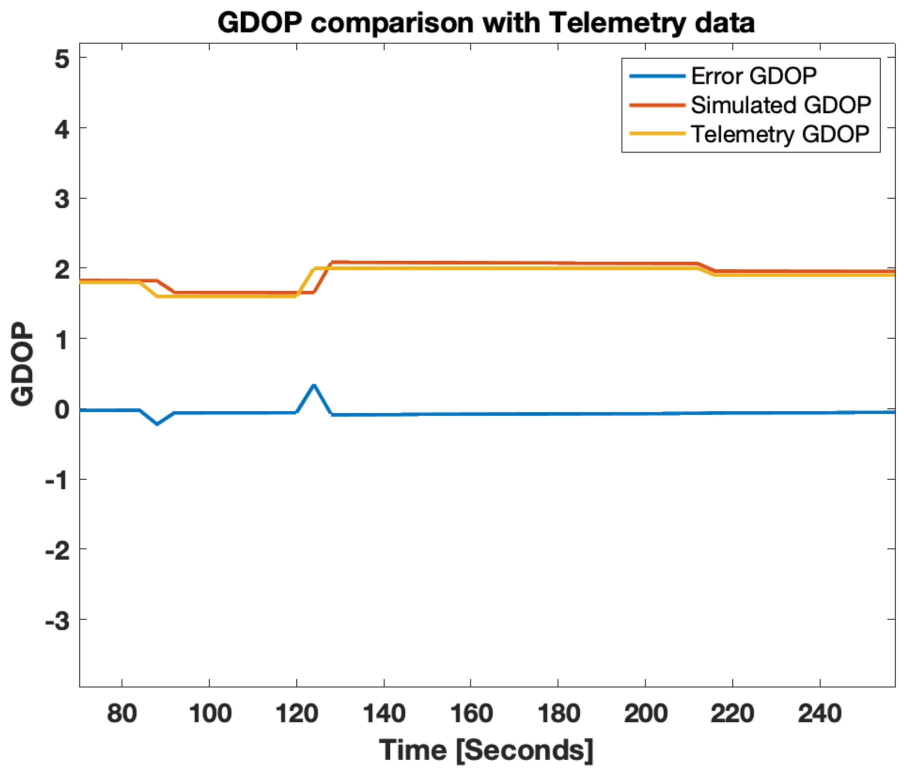


Figure 2.16. Section of Figure 2.15.

Measurement Equations

For the GNSS measurement computation, the standard formulation [28,29] has been used for pseudorange and pseudorange–rate:

$$\begin{aligned}\rho &= \|\mathbf{r}_{\mathrm{SV}}(t_{\mathrm{TX}}) - \mathbf{r}(t_{\mathrm{RX}})\| + c\Delta t_{\mathrm{R}} + c\Delta t_{\mathrm{SV}} + \Delta \mathrm{I} \\ &\quad + \Delta \mathrm{MP} + \epsilon_{\rho}, \end{aligned} \tag{2.36}$$

$$\dot{\rho} = \mathbf{e} \cdot (\dot{\mathbf{r}}_{\mathrm{SV}}(t_{\mathrm{TX}}) - \dot{\mathbf{r}}(t_{\mathrm{RX}})) + c\Delta \dot{t}_{\mathrm{R}} + \epsilon_{\dot{\rho}}, \tag{2.37}$$

where Δt_{SV} is the GPS clock bias, $\Delta \mathrm{I}$ is the ionospheric error, $\Delta \mathrm{MP}$ is the Multipath error, t_{TX} and t_{RX} are the transmission and reception epoch, respectively, c is the speed of light. The contributions of these errors have been modeled as gaussian random noises. The clock bias Δt_{R} and clock drift $\Delta \dot{t}_{\mathrm{R}}$ are modeled as a two–state stochastic dynamic model [30,31]. ϵ_{ρ} and $\epsilon_{\dot{\rho}}$ are the receiver measurement errors which represent the errors of the receiver lock loops, i.e., the Delay Lock Loop (DLL) for the code pseudorange and Frequency Lock Loop (FLL) for the pseudorange–rate. They are modeled as white gaussian noises with standard deviations computed in the following way [32]:

$$\sigma_{\mathrm{DLL}} = \left[\frac{B_{\mathrm{n}}}{2C/N_0}\left(\frac{1}{B_{\mathrm{fe}}T_{\mathrm{c}}} + \frac{B_{\mathrm{fe}}T_{\mathrm{c}}}{\pi - 1}\left(D - \frac{1}{B_{\mathrm{fe}}T_{\mathrm{c}}}\right)^2\right)\right. \tag{2.38}$$

$$\left.\left(1 + \frac{2}{T(C/N_0)(2-D)}\right)\right]^{\frac{1}{2}} \quad \text{[chips]},$$

$$\sigma_{\mathrm{FLL}} = \frac{\lambda}{2\pi T}\sqrt{\frac{4FB_{\mathrm{n}}}{C/N_0}\left[1 + \frac{1}{TC/N_0}\right]} \quad \text{[m/s]}, \tag{2.39}$$

where B_n is the closed loop bandwidth of the loops, C/N_0 computed in Hz as:

$$C/N_0 = 10\exp\left(\frac{(C/N_0)_{\mathrm{dB}}}{10}\right) \quad \text{[Hz]}, \tag{2.40}$$

B_{fe} is the front–end bandwidth of the receiver, T_{c} is the chip period, T is the integration time of the correlation process, D is the correlator spacing, λ is the wavelength of the receiver frequency, F is 1 for large C/N_0 values and 2 for C/N_0 values close to the threshold value. Note that the DLL standard deviation is expressed in chips: to convert it in meters, it has to be multiplied by the chip length which is equal to 293.05 m/chip for C/A (Coarse Acquisition) code and to 29.305 m/chip for P(Y) (Encrypted signal) code.

2.4.2 Simulation Results

State Initial Covariance P_0	Process Noise Covariance Q_0	Measurement Noise Covariance R_0
$\sigma_{\mathbf{r}} = 10_{\mathrm{m}}$	$\sigma_{\mathbf{r}} = [1 \cdot 10^{-6}\, 1 \cdot 10^{-6}\, 5 \cdot 10^{-6}]_{\mathrm{m}}$	$\sigma_{\mathrm{UERE}} = (5.2 + \sigma_{\mathrm{DLL}})_{\mathrm{m}}$
$\sigma_{\dot{\mathbf{r}}} = 0.5_{\mathrm{m/s}}$	$\sigma_{\dot{\mathbf{r}}} = [1 \cdot 10^{-7}\, 1 \cdot 10^{-7}\, 5 \cdot 10^{-7}]_{\mathrm{m/s}}$	$\sigma_{\mathrm{UERRE}} = (0.33 + \sigma_{\mathrm{FLL}})_{\mathrm{m/s}}$
$\sigma_{\Delta C_{\mathrm{d}}} = 30^{1/2}$	$\sigma_{\Delta C_{\mathrm{d}}} = 0.005 \cdot 10^{-3}$	
$\sigma_{\Delta C_{\mathrm{sp}}} = 30^{1/2}$	$\sigma_{\Delta C_{\mathrm{sp}}} = 0.005 \cdot 10^{-3}$	
$\sigma_{\Delta t_{\mathrm{R}}} = 100_{\mathrm{m}}$	$Q_{\Delta t_{\mathrm{R}},\Delta \dot{t}_{\mathrm{R}}} = \begin{bmatrix} 1.2565 \cdot 10^{-5}_{\mathrm{m}} & 5 \cdot 10^{-8} \\ 5 \cdot 10^{-8} & 1.2565 \cdot 10^{-7}_{\mathrm{m/s}} \end{bmatrix}$	
$\sigma_{\Delta \dot{t}_{\mathrm{R}}} = 100_{\mathrm{m/s}}$		

Table 2.2. Filter Parameters.

The achieved filter performances are presented in terms of position and velocity estimation accuracy with respect to the simulated trajectories. The initial conditions have been set starting from the simulated satellite initial conditions, perturbed using gaussian random errors with standard deviations in accordance to the ones collected in Table 2.2. The other states have been initialized to zero. The filter propagation time–step ΔT has been set to 0.125 seconds in order to reduce the delay between the detection of the measurement signals and the filter correction implementation.

The accuracy of the estimation is further analyzed by introducing 3σ bounds which have to be satisfied during the whole simulation. The introduction of these bounds allows to understand what is the estimation error with a confidence value of around 99.7%.

The process noise covariance of the clock bias and clock drift has been defined as a full matrix denoted by $Q_{\Delta t_{\mathrm{R}},\Delta \dot{t}_{\mathrm{R}}}$, with cross–covariance different from zero according to [30, 31, 33].

To validate the introduction of the pseudorange–rate measurements, the algorithm is compared with the approach used in [6], where only pseudorange measurements are used for POD. The same filter parameters (the ones in Table 2.2) are used for both the algorithms. The simulation time is 1 hour.

In order to obtain the simulations below, the measurement equations have been modeled by considering several error contributions: the receiver clock bias and drift errors Δt_{R} and $\Delta \dot{t}_{\mathrm{R}}$, the receiver measurement errors ϵ_ρ and $\epsilon_{\dot{\rho}}$, the ionospheric and multipath errors.

Figures 2.17 and 2.18 show the position errors with the corresponding 3σ bounds which are obtained by using only pseudorange and both pseudorange and pseudorange–rate, respectively.

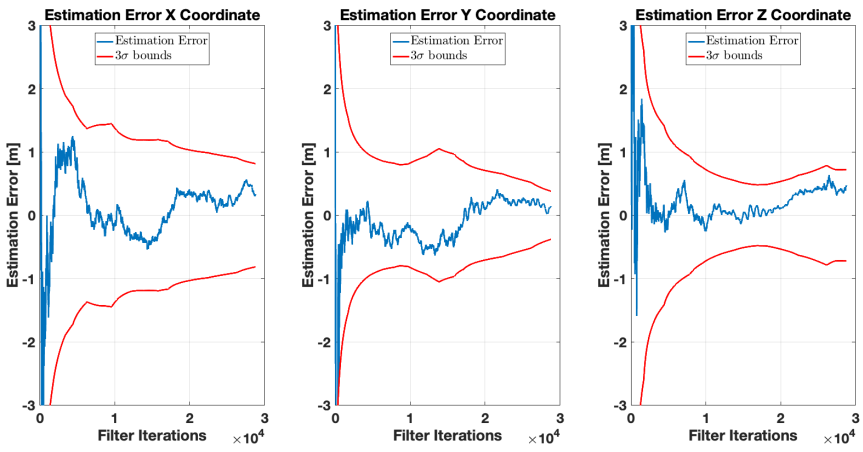


Figure 2.17. Position estimation error with pseudorange measurements and 3σ bounds.

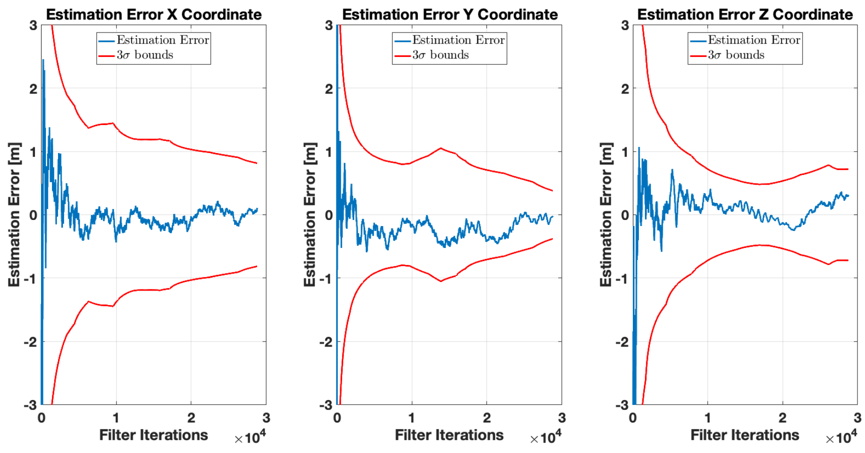


Figure 2.18. Position estimation error with both pseudorange and pseudorange–rate measurements and 3σ bounds.

It can be seen how the convergence of the algorithm is faster in the second case: there is a huge improvement for the **x** axis, whose error oscillations are minimal after 15000 iterations (about 31 minutes) while in the first case after 1 hour the error still has some peaks. The **y** component is quite similar while for the **z** component the second algorithm has a better transient in particular in the early stages of the filter.

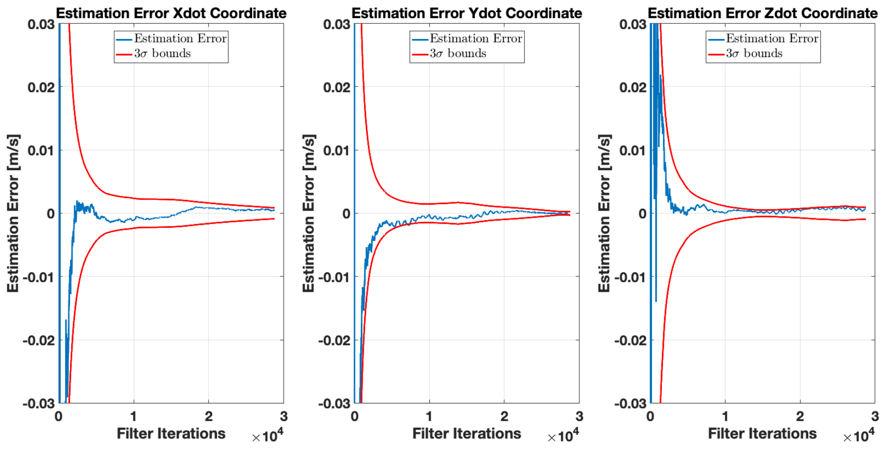


Figure 2.19. Velocity estimation error with only pseudorange measurements and 3σ bounds.

Figures 2.19 and 2.20 show the velocity estimation errors in the two cases. The difference in the convergence time is improved by the proposed algorithm: the error oscillations tend to stabilize after 5000 iterations (about 10 minutes) when both measurements are used, while in the case of a single measurement after 15000 iterations (about 31 minutes). In both cases the error stabilizes around a very small value near zero. It is important to notice that in the first case there are some

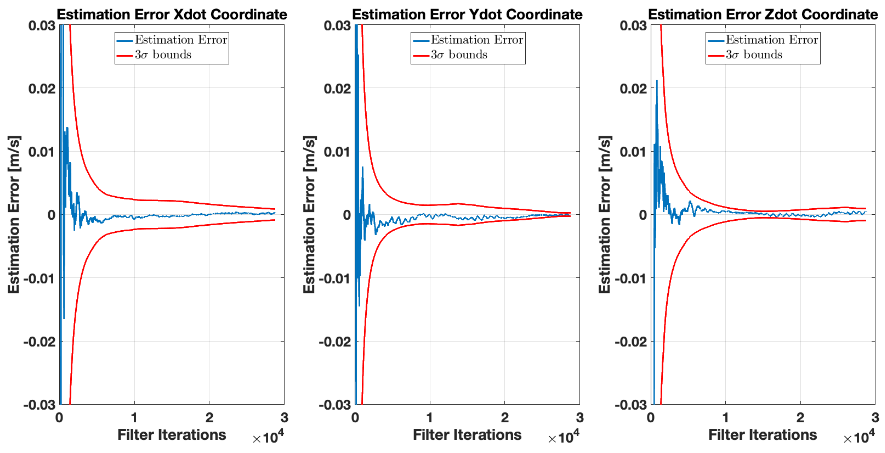


Figure 2.20. Velocity estimation error with both pseudorange and pseudorange–rate measurements and 3σ bounds.

intervals in which the error curve is out of the covariance bounds, while in the second case the error remains inside the 3σ bounds.

The same simulations have been performed by adding additional error contributions to the measurement equations, in order to verify the robustness of the filter, while maintaining the same parameters as in Table 2.2. In particular the relativistic error on the receiver clock and Shapiro effect have been added [28, 34]. The former is caused by the motion of the satellite as well as the change in the gravitational potential. These effects, caused by special and general relativity, lead to a frequency offset of the on–board clock with respect to a ground–based clock. To mitigate this effect, the satellite's clock frequencies are actually offset from their nominal values. However, non–circular satellite orbits cause deviations from the mean frequency offset which need to be considered in the on–board algorithm. The Shapiro effect instead, is a delay of the satellite signal due to the presence of the Earth's gravitational field, which causes a propagation delay due to space–time curvature.

In Figures 2.21 and 2.22 are shown the position estimation errors in the case of pseudorange and both pseudorange and pseudorange–rate measurements respectively. It can be seen how in this case the use of two measurements improves the behaviour by a great margin in particular during the transient.

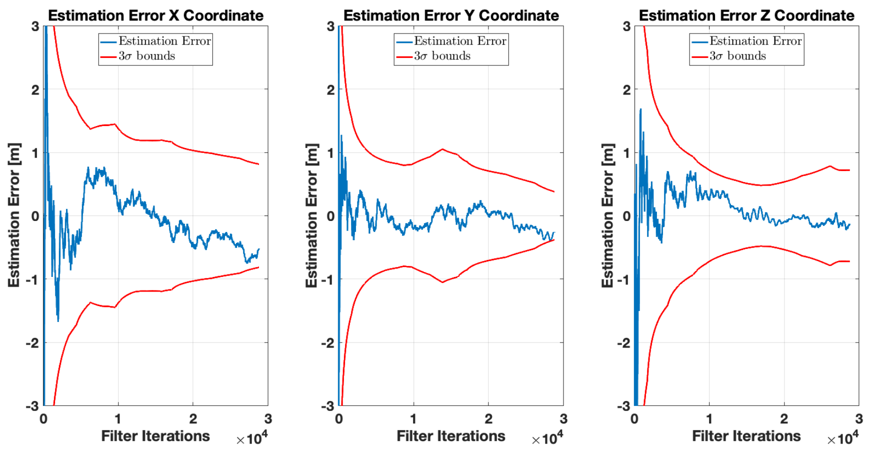


Figure 2.21. Position estimation error with pseudorange measurements and 3σ bounds considering relativistic clock errors and Shapiro Effect.

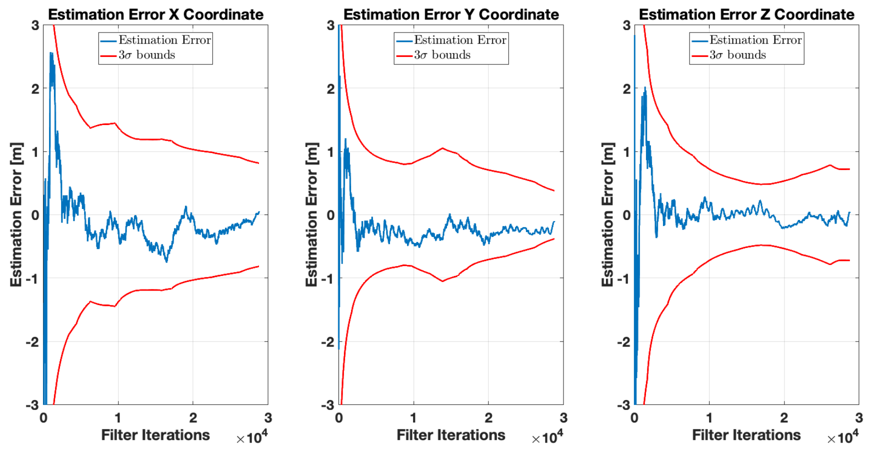


Figure 2.22. Position estimation error with both pseudorange and pseudorange–rate measurements and 3σ bounds considering relativistic clock errors and Shapiro Effect.

In Figures 2.23 and 2.24 are shown the velocity estimation errors in the case of pseudorange and both pseudorange and pseudorange–rate measurements respectively. Even in this case the use of pseudorange–rate results in an improvement of the estimation error. It can be noted how the convergence to zero is faster in the second case with less oscillations. Moreover in the first case the y coordinate slightly overcome the 3σ bounds while this behaviour is not present in the second case.

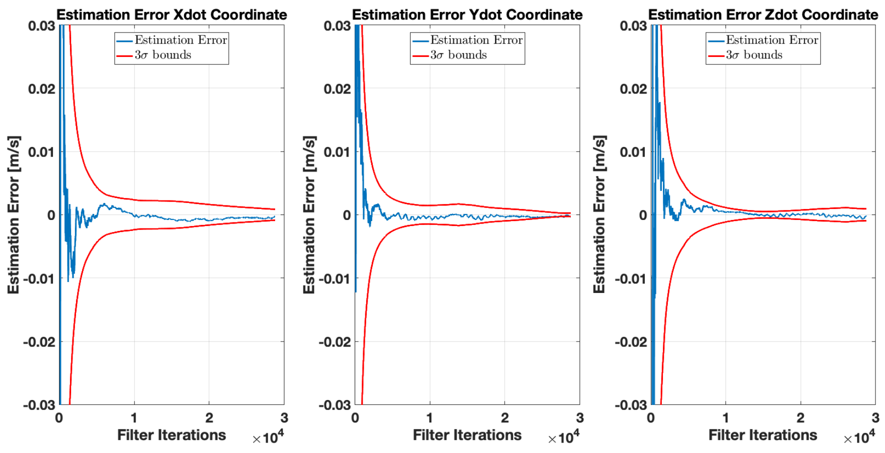


Figure 2.23. Velocity estimation error with pseudorange measurements and 3σ bounds considering relativistic clock errors and Shapiro Effect.

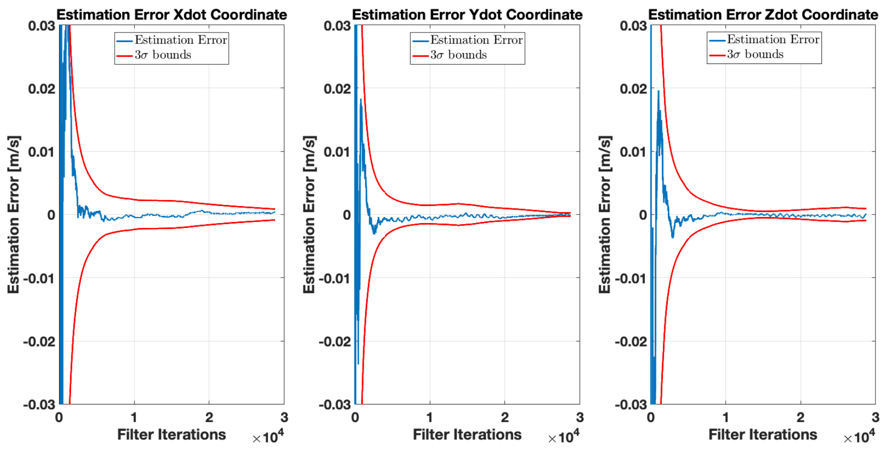


Figure 2.24. Velocity estimation error with both pseudorange and pseudorange–rate measurements and 3σ bounds considering relativistic clock errors and Shapiro Effect.

These simulations show that the proposed algorithm is able to cope with the presence of relativistic effects on the on–board clock and the presence of the Shapiro effect. The use of both observables is proven to be necessary to achieve better performances also in the presence of additional error contributions.

2.5 Future Works

The approach presented in this thesis is a simplification of the real case scenario in the fact that some quantities in the measurement equations (multipath, ionospheric delay,..) have been modeled as gaussian noises. Moreover it is not able to cope with the presence of uncertain parameters in the model as well as the CKF. A possible future scenario may consist into the implementation of actual models for these error contributions, analyzing if the EKF with pseudorange and pseudorange–rate measurements is able to maintain an acceptable level of accuracy and under which conditions.

Chapter 3

Space Qualified VPU Benchmarking of Crater Matching ODTS Solutions based on Convolutional Neural Networks

3.1 Introduction

In this section the main features characterizing the crater matching approach will be presented: an overview of different types of neural networks will be provided along with a description of the lunar environment, the different crater–based TRN approaches with an overview of the optical flow technique, which is an essential part of the TRN pipeline, will be detailed, and a description of the main state–of–the–art on–board hardwares for data processing will be given.

3.1.1 Neural Networks

Neural networks, also known as artificial neural networks (ANNs) or simulated neural networks, are a subset of machine learning and are at the heart of deep learning algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.

ANNs are comprised of node layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of

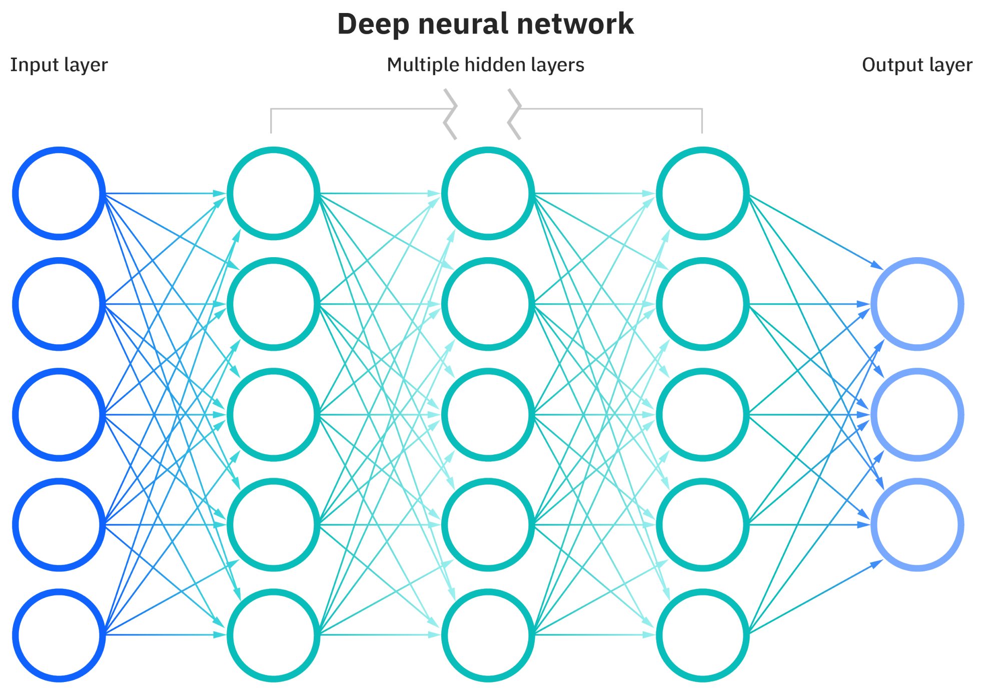


Figure 3.1. Example of an artificial neural network.

the network. Otherwise, no data is passed along to the next layer of the network. In Figure 3.1, taken from `https://www.ibm.com/it-it/topics/neural-networks`, an example of an artificial neural network architecture is shown.

Neural networks rely on training data to learn and improve their accuracy over time. However, once these learning algorithms are fine–tuned for accuracy, they are powerful tools in computer science and artificial intelligence, allowing us to classify and cluster data at a high velocity. Tasks in speech recognition or image recognition can take minutes versus hours when compared to the manual identification by human experts. One of the most well–known neural networks is Google's search algorithm.

Neural networks can be classified into different types, which are used for different purposes. Below are listed the most common types of neural networks, from which several different architectures can be derived:

- *the perceptron* is the oldest neural network, created by Frank Rosenblatt in 1958 [35]. It is an algorithm for supervised learning of binary classifiers. A binary classifier is a function which can decide whether or not an input, represented by a vector of numbers, belongs to some specific class. It is a type of linear classifier, i.e. a classification algorithm that makes its predictions based on a linear predictor function combining a set of weights with the feature vector;

- *feedforward neural networks*, or multi–layer perceptrons (MLPs). They are comprised of an input layer, a hidden layer or layers, and an output layer and are characterized by direction of the flow of information between its layers. Its flow is uni–directional, meaning that the information in the model flows in only one direction–forward–from the input nodes, through the hidden nodes (if any) and to the output nodes, without any cycles or loops [36]. While these neural networks are also commonly referred to as MLPs, it's important to note that they are actually comprised of sigmoid neurons, not perceptrons, as most real–world problems are nonlinear. Data usually is fed into these models to train them, and they are the foundation for computer vision, natural language processing, and other neural networks. Modern feedforward neural networks are trained using backpropagation method [37, 38];

- *convolutional neural networks* (CNNs) are a regularized type of feedforward neural network that learns feature engineering by itself via filters (or kernel) optimization [39]. Vanishing gradients and exploding gradients, seen during backpropagation in earlier neural networks, are prevented by using regularized weights over fewer connections. For example, for each neuron in the fully–connected layer 10.000 weights would be required for processing an image sized 100×100 pixels. However, applying cascaded convolution (or cross–correlation) kernels, only 25 neurons are required to process 5×5–sized tiles. Higher–layer features are extracted from wider context windows, compared to lower–layer features. They have applications in image and video recognition, image classification, natural language processing and in many other;

- *recurrent neural networks* (RNNs) are characterized by direction of the flow of information between its layers [40]. In contrast to uni–directional feedforward neural network, it is a bi–directional artificial neural network, meaning that it allows the output from some nodes to affect subsequent input to the same nodes. Their ability to use internal state (memory) to process arbitrary sequences of inputs makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition.

3.1.2 Lunar Environment and Crater Featuring

Moon is a challenging environment for both human survival and equipment operation. The absence of an atmosphere, the high pressures of a hard vacuum and the seismic activities are some of the factors which make the Moon a hostile environment to live and work. Moreover, the presence of craters, mountains, ridges and plains define a complex terrain that clearly complicates the correct deployment of equipment on the moon [41]. Specifically, the Moon's surface is mainly divided into large dark flat plains, craters and bright mountainous highlands. This morphology is the result of the enormous impact of asteroids and comets on the surface of the Moon that occurs with tremendous force. In particular, when large asteroids strike the surface, not only a crater is created, but also compression in and around the impact point that causes a melting of the crust. When the melted crust can no longer be compressed, it bounces back and forms a central mountain after cooling. All these peculiarities result in specific featuring of the surface, in which each area can be uniquely characterized by the high number of elements characterizing each crater. Several missions have been deployed in order to characterize the Moon surface with a high level of precision. Nowadays more than 300.000 craters have been measured and classified in several studies [42,43] and have been used to compile different databases which are becoming more and more accurate. The most used are the Head and Povilaitis database which is obtained by combining two human–generated lunar crater databases: the 5–20 km database from [44] and the > 20 km database from [45]; and the Robbins database [46] which contains approximately 1.3 billion craters with a diameter greater than about 1–2 km. The capturing is still progressing with instruments still in lunar orbit (i.e. LOLA of the Lunar Reconnaissance Orbiter [47,48]).

3.1.3 Crater Matching Techniques

The peculiarities of the lunar surface can be exploited in navigation systems which are based on feature detection over the lunar surface, namely TRN systems. These systems can be identified by using two macro criteria.

The first of them partitions the TRN systems into coordinate–relative and absolute ones, namely:

- Terrain–Relative Relative Navigation (TRRN) system, where standard optical flow and Harris corner detection algorithmic approach is adopted to evaluate tracked objects (craters/mountains) against successive acquired frames [49]. These data, together with the information about the camera specification (frame rate, spatial resolution, distance) are then used to match the features and the relative movements of the observer (the satellite platform). TRRN techniques are much easier in terms of computational complexity, being based on relatively easy–to–implement algorithms, already integrated into many frameworks and libraries with high efficiency and performance;

- Terrain–Relative Absolute Navigation (TRAN) system, differently to TRRN, applies in addition to the listed steps, a coordinate–conversion to transform relative variation of position, in absolute coordinates. This is made using a Geo–referenced feature database that has to be available as input. This database is used to perform a pattern matching between the relative sizes and distances among the tracked features, against the craters recorded on the dataset. The database includes, for each registered crater, its description, size, and depth, together with its absolute positioning. This system is indeed autonomous in the determination of absolute positions for the satellite platform. TRAN relies on more complex algorithmic systems in addition to the TRRN ones that are even more hardly compatible with most of the actual on–board flying computers.

The main algorithmic steps to implement any of TRRN and TRAN navigation systems are shown in the diagram below [50].

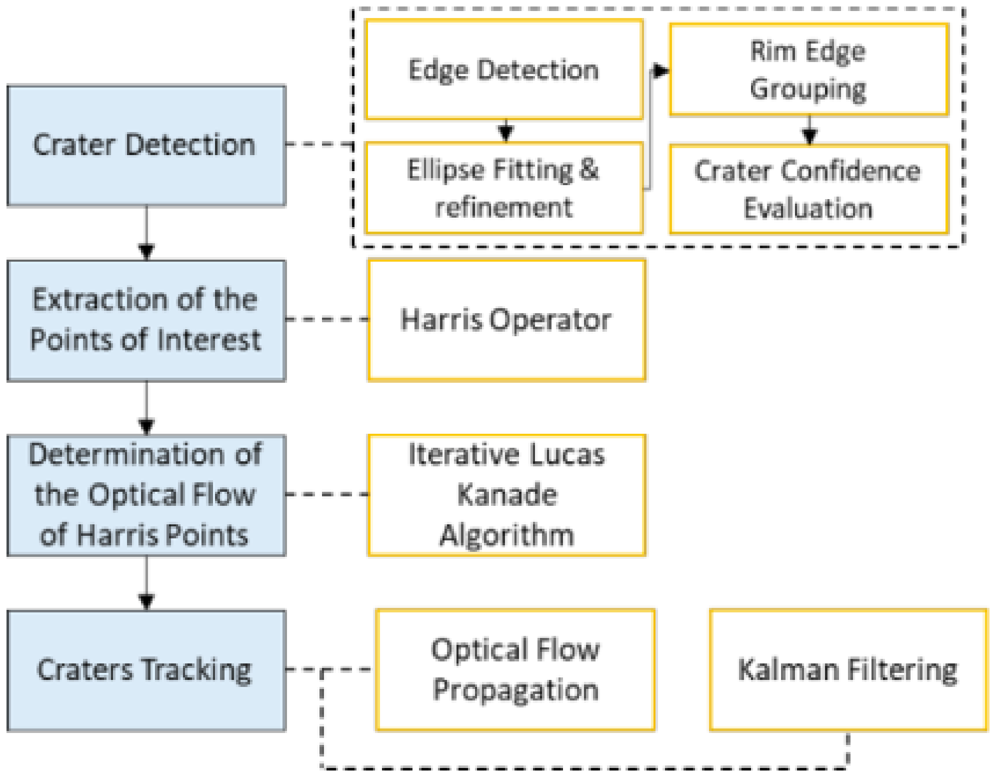

Crater Detection
Edge Detection
Rim Edge Grouping
Ellipse Fitting & refinement
Crater Confidence Evaluation
Extraction of the Points of Interest
Harris Operator
Determination of the Optical Flow of Harris Points
Iterative Lucas Kanade Algorithm
Craters Tracking
Optical Flow Propagation
Kalman Filtering

Figure 3.2. Terrain–Relative Navigation Algorithm Pipeline.

The choice between these two systems is obviously a trade–off between system complexity and required/desirable performances. The latter is strictly dependent on the user's requirements. The TRAN approach requires the handling of a huge database which needs to be available on–board and its performances are related to the accuracy and completeness of the database itself, indeed:

- missing objects in the database that are identified by the Sensor could mislead the crater matching, introducing the chance of false matching, and so additional positioning errors;
- database's record accuracy bounds the final performances of the position determination system, relaxing the constraint on the accuracy of the sensing hardware available.

The second criterion to identify TRN systems depends on the amount of prior information available on the asset to localize:

- "lost in space", so–called to represent the context in which no information at all is available on the asset's position;
- prior knowledge of the orbital position, where it's assumed that the asset already knows its position with a certain accuracy that has to be improved with crater detection and matching.

Due to the maturity and miniaturization of altimeters, star trackers and IMU, we always assume to have some of those always installed, and crater matching is to be used as a position improvement tool.

3.1.4 Optical Flow

Optical flow or optic flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer and a scene. Optical flow can also be defined as the distribution of apparent velocities of movement of brightness pattern in an image.

The concept of optical flow was introduced by the American psychologist James J. Gibson in the 1940s to describe the visual stimulus provided to animals moving through the world [51]. Gibson stressed the importance of optic flow for affordance perception, the ability to discern possibilities for action within the environment. Followers of Gibson and his ecological approach to psychology have further demonstrated the role of the optical flow stimulus for the perception of movement by the observer in the world, perception of the shape, distance and movement of objects in the world and the control of locomotion [52].

The term optical flow is also used by roboticists, encompassing related techniques from image processing and control of navigation including motion detection, object segmentation, time–to–contact information, focus of expansion calculations, luminance, motion compensated encoding, and stereo disparity measurement.

Motion estimation and video compression have developed as a major aspect of optical flow research. While the optical flow field is superficially similar to a dense motion field derived from the techniques of motion estimation, optical flow is the study of not only the determination of the optical flow field itself, but also of its use in estimating the three–dimensional nature and structure of the scene, as well as the 3D motion of objects and the observer relative to the scene, most of them using the image jacobian.

Optical flow was used by robotics researchers in many areas such as: object detection and tracking, image dominant plane extraction, movement detection, robot navigation and visual odometry [53]. Optical flow information has been recognized as being useful for controlling micro air vehicles [54].

The application of optical flow includes the problem of inferring not only the motion of the observer and objects in the scene, but also the structure of objects and the environment. Since awareness of motion and the generation of mental maps of the structure of our environment are critical components of animal (and human) vision, the conversion of this innate ability to a computer capability is similarly crucial in the field of machine vision [55].

Consider a five–frame clip of a ball moving from the bottom left of a field of vision, to the top right as the one in Figure 3.3 (`https://en.wikipedia.org/wiki/Optical_flow`). Motion estimation techniques can determine that on a two dimensional plane the ball is moving up and to the right and vectors describing this motion can be extracted from the sequence of frames. For the purposes of video compression (e.g., MPEG), the sequence is now described as well as it needs to be. However, in the field of machine vision, the question of whether the ball is moving to the right or if the observer is moving to the left is unknowable yet critical information. Not even if a static, patterned background were present in the five frames, could we confidently state that the ball was moving to the right, because the pattern might have an infinite distance to the observer.

Estimation

Sequences of ordered images allow the estimation of motion as either instantaneous image velocities or discrete image displacements. Fleet and Weiss provide a tutorial introduction to gradient based optical flow [56]. John L. Barron, David J. Fleet, and Steven Beauchemin provide a performance analysis of a number of optical flow techniques. It emphasizes the accuracy and density of measurements [57].

The optical flow methods try to calculate the motion between two image frames

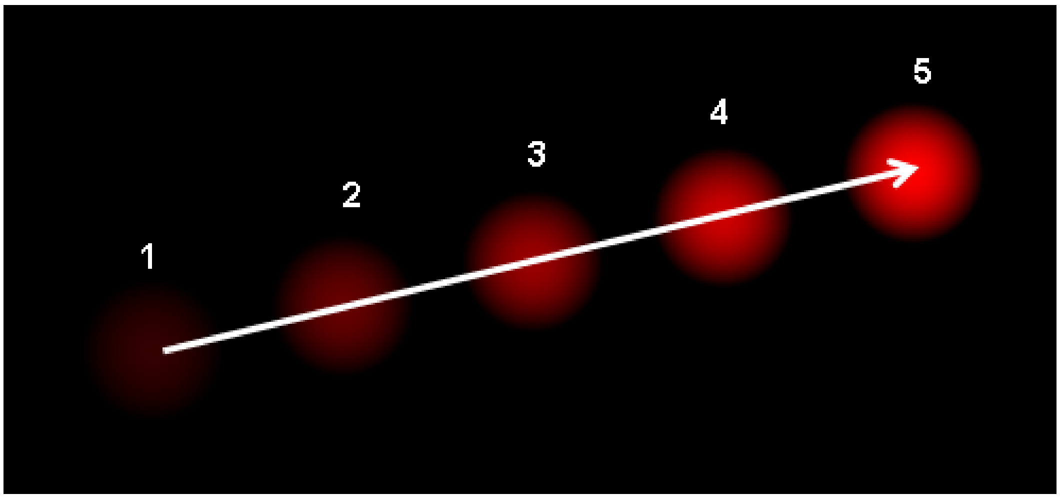


Figure 3.3. The optical flow vector of a moving object in a video sequence.

which are taken at times t and $t + \Delta T$ at every voxel[*] position. These methods are called differential since they are based on local Taylor series approximations of the image signal; that is, they use partial derivatives with respect to the spatial and temporal coordinates.

For a (2D + t)–dimensional case (3D or n–D cases are similar) a voxel at location (x, y, t) with intesity $I(x, y, t)$ will have moved by Δx, Δy and Δt between the two image frames, and the following brightness constancy constraint can be given:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t). \tag{3.1}$$

Assuming the movement to be small, the image constraint at $I(x, y, t)$ with Taylor series can be developed to get:

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial z}\Delta z + \text{h.o.t.} \tag{3.2}$$

By truncating the higher order terms (which performs a linearization) it follows that:

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t = 0 \tag{3.3}$$

or, dividing by Δt:

$$\frac{\partial I}{\partial x}\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y}\frac{\Delta y}{\Delta t} + \frac{\partial I}{\partial z}\frac{\Delta t}{\Delta t} = 0 \tag{3.4}$$

[*]In 3D computer graphics, a voxel represents a value on a regular grid in three-dimensional space. As with pixels in a 2D bitmap, voxels themselves do not typically have their position (i.e. coordinates) explicitly encoded with their values. Instead, rendering systems infer the position of a voxel based upon its position relative to other voxels (i.e., its position in the data structure that makes up a single volumetric image).

which results in:

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial z} = 0 \tag{3.5}$$

where V_x and V_y are the x and y components of the velocity or optical flow of $I(x, y, t)$ and $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$ and $\frac{\partial I}{\partial t}$ are the derivatives of the images at (x, y, t) in the corresponding directions. I_x, I_y and I_t can be written for the derivatives in the following. Thus:

$$I_xV_x + I_yV_y = -I_t \tag{3.6}$$

or:

$$\nabla I\mathbf{V} = -I_t \tag{3.7}$$

where $\mathbf{V} = (V_x, V_y, 0)^T$. This is an equation in two unknowns and cannot be solved as such. This is known as the aperture problem of the optical flow algorithms. To find the optical flow another set of equations is needed, given by some additional constraint. All optical flow methods introduce additional conditions for estimating the actual flow.

Methods for determination

There are several methods which can be used to solve the previous problems. Below some of the most used are presented. Many of these, in addition to the current state–of–the–art algorithms are evaluated on the Middlebury Benchmark Dataset [58].

- *Phase correlation* – inverse of normalized cross-power spectrum [59];
- *Block-based methods* – minimizing sum of squared differences or sum of absolute differences, or maximizing normalized cross–correlation [60];
- Differential methods of estimating optical flow, based on partial derivatives of the image signal and/or the sought flow field and higher–order partial derivatives, such as:
  - *Lucas–Kanade method* – regarding image patches and an affine model for the flow field [61];
  - *Horn–Schunck method* – optimizing a functional based on residuals from the brightness constancy constraint, and a particular regularization term expressing the expected smoothness of the flow field [62];
  - *Buxton–Buxton method* – based on a model of the motion of edges in image sequences [63];
  - *Black–Jepson method* – coarse optical flow via correlation [64];

  - *General variational methods* – a range of modifications/extensions of Horn–Schunck, using other data terms and other smoothness terms [65].

- *Discrete optimization methods* – the search space is quantized, and then image matching is addressed through label assignment at every pixel, such that the corresponding deformation minimizes the distance between the source and the target image. The optimal solution is often recovered through Max–flow min–cut theorem algorithms, linear programming or belief propagation methods [66, 67].

3.1.5 On–Board Data Processing and Neural Networks

Satellite on–board data processing requires a balance between low power consumption and high performance in terms of robustness and re–configurability. The challenge associated with TRRN and TRAN pertains to the considerable computational complexity that must be managed directly on the on–board system.

The field of on–board computing is experiencing rapid growth, with several processors that meet the standards for use in space becoming increasingly accessible in the commercial market. In particular, novel parallel computing structures, rooted in GPU and VPU principles, are emerging as viable options for on–board processing. Additionally, a specialized class of AI accelerator microprocessors is being developed to enhance the efficiency of machine vision operations in energy–efficient settings. These microprocessors excel in executing image processing tasks while in orbit and are currently undergoing thorough environmental testing conducted by ESA. Laboratory experiments have extensively tested AI techniques and neural networks to tackle the demanding crater identification task, which remains the most critical challenge among TRN algorithms.

Given its foundation in computer vision, on–board spacecraft, efficient image processing can be achieved by employing an appropriate CNNs. The strength and popularity of CNNs lie in their capacity to automatically extract features through training, enabling effective differentiation between various classes. When properly trained on a relevant dataset, these types of neural networks excel in capturing significant contextual features from images [68]. This proficiency ensures system robustness against variations in acquisition angles, lighting conditions, and camera–induced distortions, thereby addressing some of the limitations seen in classical crater detection algorithms.

Leading the charge in the AI industry, influential names like Intel, Google, and Nvidia are driving the advancement of edge AI. They are achieving this by offering hardware platforms and accelerators that come in compact forms. Among these, one of the most notable is the Intel Movidius Myriad, which has garnered attention for its integration and successful testing within ESA's Phisat-1 satellite. Another standout player is the Nvidia Jetson Nano. Below a brief description of these hardware accelerators:

- "Movidius Stick" (see Figure 3.4, taken from `https://www.intel.com/content/www/us/en/developer/articles/tool/neural-compute-stick.html`) – The Intel Movidius NCS stands as a compact, fanless USB drive designed for instructive AI programming exploration. Empowered by the Movidius Visual Processing Unit, a paragon of power efficiency and high performance, this device fuels its capabilities. Its core is embedded with an Intel Movidius Myriad X VPU. Within its architecture lies the capability to seamlessly transform foundational CNN architecture from its original format (such as TensorFlow or Torch models) into the OpenVINO framework. This conversion prowess is harnessed through the utilization of the OpenVINO library. Moreover, the Movidius Stick expedites the inference process for deep learning co- processors integrated into USB sockets. Notably, the toolkit supports diverse execution across computer vision accelerators, encompassing GPU, CPU, and FPGA, via a standardized API. Additionally, it enables on–the–edge deep learning inference, contributing to a comprehensive AI programming experience [69];

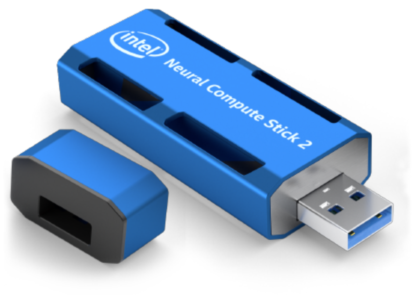


Figure 3.4. Myriad Compute Stick.

- "Jetson TX2" (see Figure 3.5, taken from `https://developer.nvidia.com/embedded/jetson-tx2`) – In the realm of deep learning inference, the Jetson TX2 model boasts twice the energy efficiency of its predecessor, the Jetson TX1, and outperforms Intel's Xenon server CPU. This heightened level of efficiency fundamentally shifts the landscape, making it feasible to transition advanced AI processing from centralized cloud setups to the edge. Facilitated by resources such as LSTMs, RNNs, the TensorRT library, and Nvidia's CUDA Deep Neural Network library (cuDNN), the Jetson TX2 significantly expedites the advancement of cutting–edge Deep Neural Network designs.

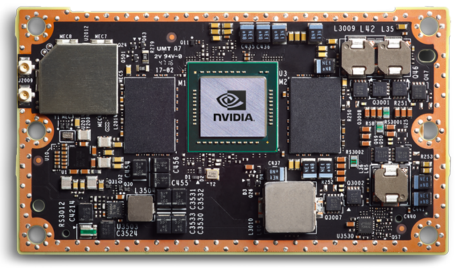


Figure 3.5. Jetson TX2 board.

3.2 State of the Art and Original Contribution

In recent years, the moon is gaining back attractiveness in terms of human exploration for scientific purposes, for resources exploitation and as a baseline for Mars exploration. A lot of research has been focused on developing OD–based navigation algorithms able to cope with real-time and local constraints with lack or limited access to standard measurements, e.g. ground tracking station measurements from Earth. The ability of the satellite to self-estimate its position, aided by on–board sensor measurements and auxiliary observables, is a core feature to achieve in an environment like the Moon, where ground-control monitoring is not yet available.

In this context crater matching, that is matching features over impact crater images, represents an interesting technique for boosting ODTS algorithms. This approach requires intense computational resources and on-board storage, which would require the direct acquisition, data processing and position determination performed on–board, together with the storage of both the available crater databases and of the continuously acquired images.

In [70] the authors develop a Crater Matching and Detection Algorithm (CDMA) which makes use of a combination of TRRN and TRAN features. The crater detection is achieved by pairing shadowed and illuminated objects of the image having a similar size and a relative orientation consistent with the direction of the light, while crater matching uses the parameters of the detected craters to locate them into a Geo–referenced database. Moreover common features are tracked over successive frames by using Harris Corner tracking [49].

In recent years, AI has come out as one of the most attractive solutions to the crater matching problem. [71] proposed the use of an U–Net style CNN to detect the crater features in the moon images which allows segmentation at pixel–level resolution [72]; [73, 74] proposed an improvement of the previous approach by training the network to detect craters also under different illumination conditions. [75] proposed the same methodology to tackle the problem of crater matching on Mars' craters.

In [76] the authors proposed the use of mixed unsupervised and supervised learning to detect craters by using multiscale Hypothesis Generation (HG) step [77, 78], followed by Hypothesis Verification (HV) step.

The use of AI allows to decrease the computational load of the CDMA and can be effectively employed in an end–to–end ODTS pipeline. In [50] the authors proposed an ODTS scheme based on TRAN approach to track a satellite position in a Elliptical Lunar Frozen Orbit (ELFO). In this work, the crater matching task has been performed by using the U–Net architecture. [79] proposed an TRAN scheme for navigation called FederNet in which a Mask R–CNN architecture is used as a feature extractor [80].

Crater matching algorithms have also been applied to the Moon landing problem. In [81] the authors proposed the use of an object detection network in order to detect directly the crater coordinates in the image frame, without the need for further processing. This approach was able to speed–up considerably the whole TRN pipeline.

In this work a detailed analysis of several AI–based crater detection approaches have been carried out. In particular a benchmarking of different crater matching techniques has been performed in order to verify their performances and applicability in the field of real–time orbit determination. To do so, we started by performing a trade–off to select the most promising neural network architecture for the purpose of crater matching OD technique. The benchmarking is performed by using a space–qualified processor which is characterized, in addition to a CPU and an FPGA, by a VPU built for running AI applications.

3.3 Theory and calculations

In this section the benchmarked neural network architectures are presented along with the training procedure and the key performance indicators considered in the analysis. The benchmark platform and its specifications will also be described in detail.

3.3.1 Methods

In this work, the focus of the benchmarking is on the *crater detection* part of the TRN pipeline. This is one of the most computationally involved part of the overall TRN system; indeed, neural networks are used to take over the overly complex algorithms for feature detection and extraction. The choice of the neural network is of utmost importance for determining the performances and the constraints of the on–board navigation algorithm. The former relates to the estimation rate, i.e. the time between one position estimate and the next one, and also to the precision in the estimate itself; the latter are usually related to the instruments equipped on the satellite (sensors, processor capabilities, ...). Choosing a heavy network may result in a better feature extraction process but it usually takes a long time to perform the inference and this can lead to accumulated delays in the overall process; choosing a smaller network allows to avoid long time delays due its faster inference time but can also result in slightly worse features.

Benchmarking will be based on several performance parameters described in Table 3.1:

Parameter	**Description**
Input Precision	Bit precision of the model input (float16, float32, uint8).
Read model time	Amount of time taken to read the model files.
Reshape model time	Amount of time to resize the model to match the image size if needed.
Compile model time	Amount of time needed to load the model on the inference device (CPU, Myriad).
First inference time	Amount of time to perform the first inference.
Latency	Amount of time it takes to process a single inference request.
Throughput	Number of inferences per second the device can perform.

Table 3.1. Benchmarking Performance Parameters.

In addition to the performance parameters described above, in order to select the most appropriate NN model architecture, it is useful to introduce other qualitative parameters, which are useful in order to best perform a trade-off analysis. In particular, the *output feature* is introduced as a parameter, i.e. what information is output and how it is organized, and the *tile dimension*, which brings a more univocal pattern of craters as the size increases, as more craters will be considered.

In this paper, three different neural networks are considered:

- the U–Net [71, 74] is a deep convolutional neural network used for segmentation tasks originally introduced to perform semantic segmentation of medical images (e.g. brain images segmentation and liver images segmentation) [82]. Its architecture is shown in Figure 3.6. The network consists of a contracting path and an expansive path, which gives it the u–shaped architecture. The contracting path is a typical convolutional network that consists of repeated application of convolutions, each followed by a rectified linear unit (ReLU) [83] and a max pooling operation. During the contraction, the spatial information is reduced while feature information is increased. The expansive pathway combines the feature and spatial information through a sequence of up–convolutions and concatenations with high–resolution features from the contracting path. The main idea is to supplement a usual contracting network by successive layers, where pooling operations are replaced by upsampling operators. Hence these layers increase the resolution of the output. A successive convolutional layer can then learn to assemble a precise output based on this information. One important modification in U–Net is that there are a large number of feature channels in the upsampling part, which allow the network to propagate context information to higher resolution layers. As a consequence, the expansive path is more or less symmetric to the contracting part, and yields a u–shaped architecture. The network only uses the valid part of each convolution without any fully connected layers. To predict the pixels in the border region of the image, the missing context is extrapolated by mirroring the input image. This tiling strategy is important to apply the network to large images, since otherwise the resolution would be limited by the GPU memory. This network takes as input a crater image and performs edge detection on the craters. The output, as shown in Figure 3.6, is an image in which the edges of the craters are extrapolated from the rest of the background;

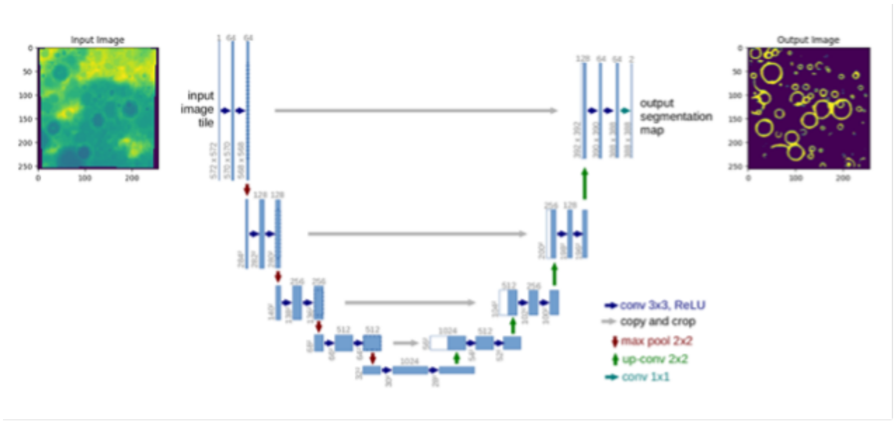


Figure 3.6. U–Net architecture.

- MobileNetV2/MobileNetV3 with Single–Shot Detector (SSD). The architecture is similar to the one in Figure 3.7 with the MobileNetV2 or MobileNetV3 as a backbone instead of the classical VGG16 [84]. MobileNet stands as the predominant CNN–based model utilized in smartphone applications and embedded devices thanks to its compact dimensions and efficient performance. Its concept revolves around the employment of depthwise separable convolution to construct a lightweight model that demands lower computational resources. Traditional CNN architectures often involve convolution to be carried out point–to–point between the image and filter, leading to heightened computational demands. MobileNet's innovation lies in its depthwise convolution, wherein a single filter is used per input channel. Subsequently, pointwise convolution employs a 1×1 convolution to combine the output from the depthwise layer linearly. This strategic separation of convolutions significantly reduces both computation requirements and model size. This architecture serves as the foundational "base network" within the broader model structure, recognized as a standard framework esteemed for its excellence in high–quality image classification.

  The architectures used in this work are the evolutions of the MobileNet architecture in terms of layer efficiency [85]. In particular the MobileNetv2 [86] introduced the linear bottleneck and inverted residual structure in order to make even more efficient layer structures by leveraging the low rank nature of the problem. This structure is defined by a 1×1 expansion convolution followed by depth–wise convolutions and a 1×1 projection layer. The input and output are connected with a residual connection if and only if they have the same number of channels. This structure maintains a compact representation at the input and the output while expanding to a higher–dimensional feature space internally to increase the expressiveness of nonlinear per-channel transformations.

  The MobileNetv3 [87] further enhanced the previous version by adding the Squeeze–and–Excite operation [88] in the residual layer. Moreover a new activation function called *swish* [89–91] has been introduced as a drop–in replacement for the ReLU that significantly improves the accuracy of neural networks.

  To culminate the model, the SSD architecture is incorporated in its latter portion [84]. The SSD technique employs a feed-forward convolutional network, generating a predetermined array of bounding boxes and associated scores indicating the presence of object class instances within these boxes. This output is then refined through a non-maximum suppression process, ultimately yielding the conclusive detections.

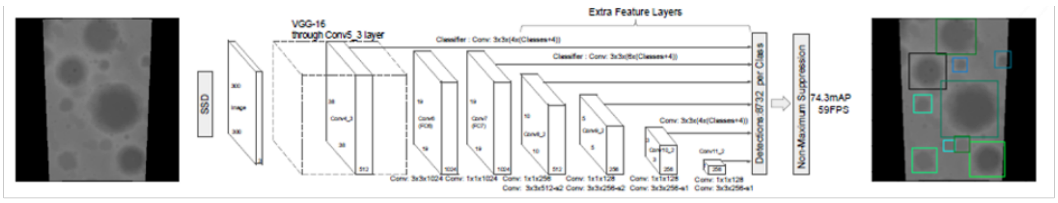
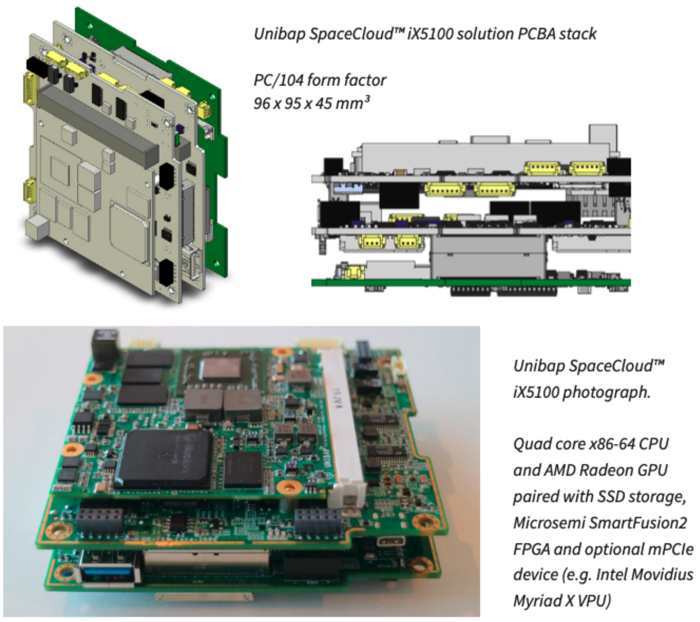


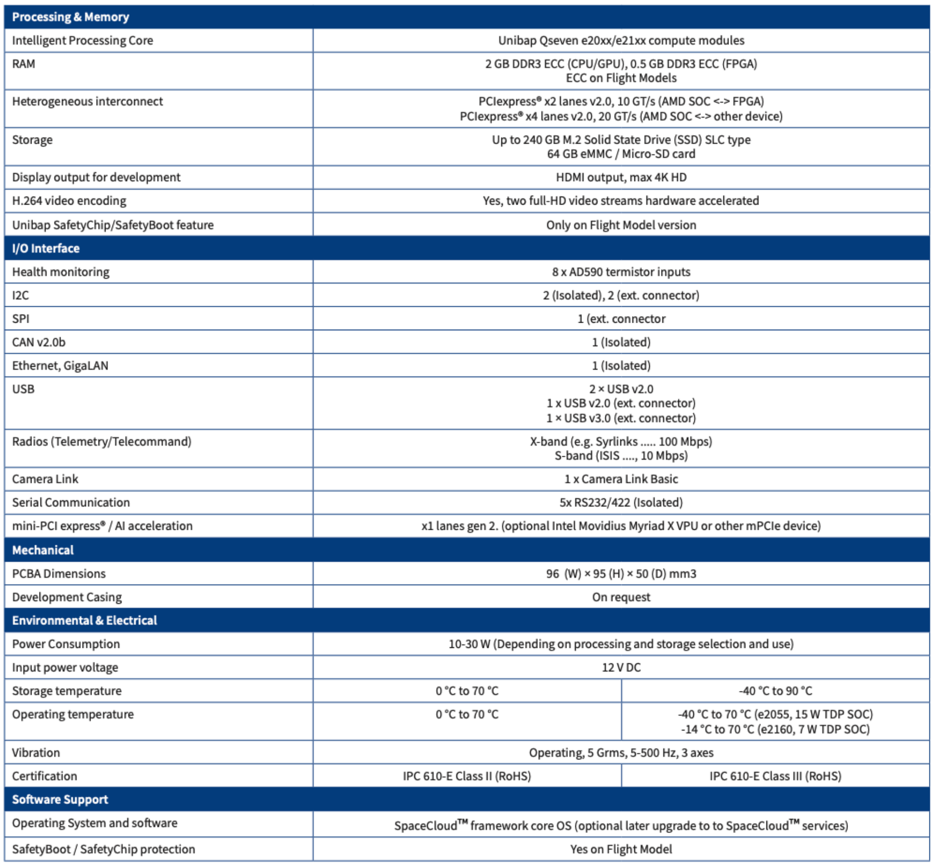
Figure 3.7. SDD network architecture. The original feature extractor is based on VGG-16.

It is worth noticing how both these neural networks could achieve detection performances compatible for the sake of achieving a robust observable as will be described in Section 3.3.3; the considerations about the quality of such observable and how these impact the overall OD system are not presented here, while we have been focusing in this work on the compatibility of this networks to be executed efficiently on dedicated HW platforms.

3.3.2 Hardware Specifications


Unibap SpaceCloud™ iX5100 solution PCBA stack
PC/104 form factor
96 x 95 x 45 mm³
Unibap SpaceCloud™ iX5100 photograph.
Quad core x86-64 CPU and AMD Radeon GPU paired with SSD storage, Microsemi SmartFusion2 FPGA and optional mPCIe device (e.g. Intel Movidius Myriad X VPU)
Figure 3.8. Unibap iX5–100 Board.

The three models outlined in Section 3.3.1 serve as testing benchmarks for evaluating the inference performance during the deployment of models onto edge hardware. Conducting these benchmarks on an on–the–edge device introduces added memory constraints, given that memory and processor capacities are notably smaller than those found in standard workstations. The experiment is performed by using Unibap's iX5–100 (see Figure 3.8) [92], a SpaceCloud® computer solution for large and small spacecraft. The iX5–100 uses computer modules with support for GPU and one or more neuron network accelerators. Moreover, the iX5–100 can be customized in several steps since the standard I/O card contains in/outputs for camera link, stepper motor control, LVDS for radio communication, AD590 temperature measurement, and a mini–PCIe slot for calculation expansion. All hardware specifics are shown in Figure 3.9 [92]. The iX5–100 board features the Intel Movidius Myriad X VPU capable of performing fast inferences and whose processor speed up machine learning models by using the inference engine provided by the Intel OpenVINO Toolkit.



Processing & Memory		
Intelligent Processing Core	Unibap Qseven e20xx/e21xx compute modules	
RAM	2 GB DDR3 ECC (CPU/GPU), 0.5 GB DDR3 ECC (FPGA) ECC on Flight Models	
Heterogeneous interconnect	PCIexpress® x2 lanes v2.0, 10 GT/s (AMD SOC <-> FPGA) PCIexpress® x4 lanes v2.0, 20 GT/s (AMD SOC <-> other device)	
Storage	Up to 240 GB M.2 Solid State Drive (SSD) SLC type 64 GB eMMC / Micro-SD card	
Display output for development	HDMI output, max 4K HD	
H.264 video encoding	Yes, two full-HD video streams hardware accelerated	
Unibap SafetyChip/SafetyBoot feature	Only on Flight Model version	
I/O Interface		
Health monitoring	8 x AD590 termistor inputs	
I2C	2 (Isolated), 2 (ext. connector)	
SPI	1 (ext. connector	
CAN v2.0b	1 (Isolated)	
Ethernet, GigaLAN	1 (Isolated)	
USB	2 × USB v2.0 1 x USB v2.0 (ext. connector) 1 × USB v3.0 (ext. connector)	
Radios (Telemetry/Telecommand)	X-band (e.g. Syrlinks ..... 100 Mbps) S-band (ISIS ...., 10 Mbps)	
Camera Link	1 x Camera Link Basic	
Serial Communication	5x RS232/422 (Isolated)	
mini-PCI express® / AI acceleration	x1 lanes gen 2. (optional Intel Movidius Myriad X VPU or other mPCIe device)	
Mechanical		
PCBA Dimensions	96 (W) × 95 (H) × 50 (D) mm3	
Development Casing	On request	
Environmental & Electrical		
Power Consumption	10-30 W (Depending on processing and storage selection and use)	
Input power voltage	12 V DC	
Storage temperature	0 °C to 70 °C	-40 °C to 90 °C
Operating temperature	0 °C to 70 °C	-40 °C to 70 °C (e2055, 15 W TDP SOC) -14 °C to 70 °C (e2160, 7 W TDP SOC)
Vibration	Operating, 5 Grms, 5-500 Hz, 3 axes	
Certification	IPC 610-E Class II (RoHS)	IPC 610-E Class III (RoHS)
Software Support		
Operating System and software	SpaceCloud™ framework core OS (optional later upgrade to to SpaceCloud™ services)	
SafetyBoot / SafetyChip protection	Yes on Flight Model	

Figure 3.9. Unibap's iX5–100 Hardware Specifications.

3.3.3 Algorithm Implementation

In this section the training and validation procedure of the models will be defined along with a brief description of the benchmarking algorithm implementation.

Training Procedure

The input data were generated by randomly cropping digital elevation map (DEM) images from the Lunar Reconnaisance Orbiter (LRO) and Kaguya merged digital elevation model, which spans ± 60 degrees in latitude and the full range in longitude, and has resolution of 512 pixels/degree (or 59 m/pixel) [93, 94]. The global greyscale map is a Plate Carree projection with a resolution of 184320×61440 pixels and with a bit depth of 16 bits/pixel. The map has been downsampled to 92160×30720 pixels with a bit depth of 8 bits/pixel. The use of an elevation map, instead of an optical one, makes it easier to train the neural network. The absence of effects related to the sunlight direction, indeed, reduces possible variations in the appearance of the craters.

Each input image has been generated by following a four–step procedure:

1. randomly cropping a square area of the global map. The position of this region has been selected with a uniform distribution and its length has been derived from a log–uniform distribution with minimum and maximum bounds of 500 and 6500 pixels (equivalent to 59 and 770 km respectively) in order to catch both small and large craters;

2. downsampling the cropped image to 256×256 pixels for the U–Net model and to 320×320 pixels for the MobileNet models;

3. transforming the image to an ortographic projection using the Cartopy Python package in order to minimize image distortion. This transformation often produces non–square images padded with zeros (see Figure 3.7);

4. rescaling image intensity to boost contrast.

Image normalization within the range of [0,1] has been applied. The ground–truth data for the U–Net are generated for each input image by encoding craters as rings with thicknesses of 1 pixel and with centers and radii derived from the combined Head and Povilaitis dataset [44, 45]. While the ground–truths for the MobileNet architectures are characterized by the craters' bounding–boxes coordinates.

The input data have been divided into training and validation and test set. Each of them sampled from equal–sized and mutually exclusive portions of the Moon, spanning the full longitude range. Each dataset contains 15000 tiles of images. The three different ANN structures explained in Section 3.3.1 are trained in order to understand which is the best architecture that led to extracting crater coordinates. The training hyperparameters are specified in Table 3.2. The U–Net model has been trained for 4 epochs, while the other ones have been trained for 150 epochs.

Hyperparameters	U–Net	MobileNetV2/V3 with SSD
Learning Rate	1e-4	5e-4
Batch Size	8	32
Optimizer	Adam	AdamW
Loss Function	Binary Cross Entropy	SmoothL1 + Cross Entropy

Table 3.2. Hyperparameters Specifications.

The accuracy of the models has been validated by using the *Precision* and *Recall* metrics on the test set. The results show similar performances between the models with respect to these two parameters: around 0.6 for the former and 0.9 for the latter. These values of accuracy are sufficient in order to produce a robust observable. It is not necessary to have a model able to detect all the possible craters in an image since only a subset of them, possibly well distributed, should be sufficient in order to determine the position of the satellite if paired with other sensors measurements like IMU and altimeter data. Moreover, in the case of a TRAN approach, the higher the number of craters, the higher is the computational effort due to the database search.

SpaceCloud Testing

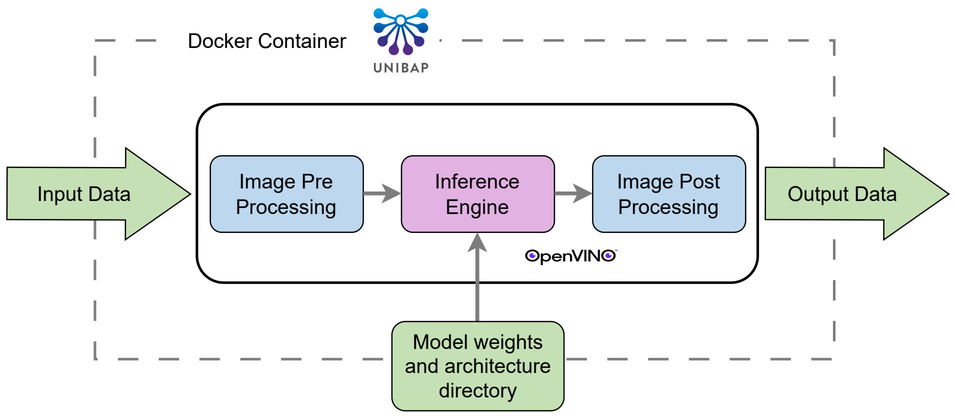


Figure 3.10. Inference Workflow.

To test the above models over the space–qualified hardware described in Section 3.3.2, model weights must be converted in a suitable format compatible with the Intel Movidius Myriad X VPU. This procedure is characterized by several steps:

1. model conversion from PyTorch or TensorFlow to ONNX format;

2. model optimization by scaling the model weights to FP16;
3. model conversion to IR format;
4. final xml and bin files generation.

At this point, model inference is executed over the Unibap iX5–100 SpaceCloud platform to conduct benchmarking and assess model performance, determining the configuration that best aligns with the camera's frame rate. The benchmarking algorithm has been deployed on the platform by means of a Docker container including the OpenVINO framework and all the necessary packages (see Figure 3.10). This allows the algorithm to be portable over different platforms supporting the framework. Final results will be presented in the next section.

3.4 Results and Discussion

This section presents some of the benchmarking results that have been ran over four kinds of configurations, specifically:

- U–Net model tested on Unibap CPU;
- U–Net model tested on Unibap Myriad;
- MobileNet models tested on Unibap CPU;
- MobileNet models tested on Unibap Myriad.

The different results have been obtained by performing inferences over a 60 seconds time interval. Different input shapes and input precisions have also been considered. In particular for the U–Net the input images are 256×256 in grey scale; for the MobileNetV2 and MobileNetV3 the input images are 320×320 in RGB format. The results are presented in Tables 3.3–3.5.

Inference Hardware	CPU	CPU	Myriad	Myriad
Input Precision	FP16	FP32	FP16	FP32
Read model time (msec)	81.73	91.74	76.69	79.04
Reshape model time (msec)	0.27	0.32	0.27	0.27
Compile model time (msec)	574.81	611.12	12920.61	13018.27
First inference time (msec)	9344.02	9908.14	509.84	509.87
Avg latency (msec)	9319.93	9425.47	506.89	507.52
Min latency (msec)	9262.18	9258.31	505.63	506.54
Max latency (msec)	9436.71	9759.30	507.85	508.41
Throughput (frame/s)	0.11	0.11	1.97	1.97

Table 3.3. U–Net Performance Parameters.

Inference Hardware	CPU	CPU	CPU	Myriad
Input Precision	uint8	FP16	FP32	uint8
Read model time (msec)	170.11	168.14	169.27	170.92
Compile model time (msec)	1118.31	1089.33	1083.36	12068.99
First inference time (msec)	140.39	141.00	135.78	50.37
Avg latency (msec)	107.06	111.50	107.85	48.45
Min latency (msec)	105.53	107.39	106.93	47.98
Max latency (msec)	138.96	310.05	143.24	49.15
Throughput (frame/s)	9.31	8.94	9.24	20.52

Table 3.4. MobileNetV2 Performance Parameters.

With the data presented in the tables, it is possible to compare the neural network models examined based on the hardware; this information is considered relevant (among many others) to analyze the compatibility in being used in a complete software suite for orbit determination.

To facilitate comparison, a summary table is given, trying to highlight the most important parameters identified in the Section 3.3.1 compared for each NN model and with the same hardware (see Table 3.6).

The most important parameter for the application of crater matching techniques is the number of analyzable crater images frames per second, namely the throughput. High throughput is fundamental in real–time scenarios in which a large amount of data needs to be inferenced simultaneously. This parameter is compared with the

Inference Hardware	**CPU**	**CPU**
Input Precision	FP16	FP32
Read model time (msec)	575.98	576.29
Reshape model time (msec)	0.28	0.27
Compile model time (msec)	1933.23	1996.34
First inference time (msec)	101.83	123.47
Avg latency (msec)	72.72	75.72
Min latency (msec)	71.81	70.23
Max latency (msec)	101.74	194.25
Throughput (frame/s)	13.69	13.14

Table 3.5. MobileNetV3 Performance Parameters.

frame rate of the space–based camera taken as a reference. The camera taken as reference for the purpose of this analysis achieves performances similar to the one used in [95], with a frame rate of around 10 fps. Indeed, it's possible to see how the achieved performances actually meet and exceed this threshold value, allowing for real–time processing and future OD measurements in reduced time windows.

Also relevant is latency which, as described in Section 3.3.1, quantifies the time required to process the single inference request with data percolating from input to output the processing pipeline.

NN Model	HW with input precision	Avg Latency (msec)	Throughput (frame/s)	Output Feature	Tile Dimension
U–Net	CPU(FP16)	9319.93	0.11	Detected matrix mask	256×256
	CPU(FP32)	9425.47	0.11	Detected matrix mask	256×256
	Myriad(FP16)	506.89	1.97	Detected matrix mask	256×256
	Myriad(FP32)	507.52	1.97	Detected matrix mask	256×256
MobileNetv2 with SSD	CPU(uint8)	107.06	9.31	Matrix Crater coordinates	320×320
	CPU(FP16)	111.50	8.94	Matrix Crater coordinates	320×320
	CPU(FP32)	107.85	9.24	Matrix Crater coordinates	320×320
	Myriad(uint8)	48.45	20.52	Matrix Crater coordinates	320×320
MobileNetV3 with SSD	CPU(FP16)	72.72	13.69	Matrix Crater coordinates	320×320
	CPU(FP16)	75.72	13.14	Matrix Crater coordinates	320×320
	Myriad	Not Supported	Not Supported	Not Supported	Not Supported

Table 3.6. Summary of Performance Results.

From these few shown results, already many are the considerations that could be made. Indeed, for the performance found on all types of hardwares, it is evident that U–Net is an NN model that could hardly be adopted for crater matching applications, specifically in the field of Orbit determination; even if ran over an optimized hardware platform, such as Myriad, the achievable frame rate is not sufficient to satisfy the key requirement of frame rate alignment with the spatialized camera. Furthermore, high average latency times occur. To these times it must also be added that the output feature of the network, in order to obtain usable information, must be further being processed, introducing additional delay components. The MobileNetV2 + SSD model instead, exhibits an adequate frame rate for applications when combined with the Myriad board. It should be noted that the use of the Myriad results in approximately 100% better performances. Even when considering latency times and output features organized and ready for subsequent processing steps, and tiles of an interesting size for recognition applications, the MobileNet V2 + SSD appears to be a good candidate model to be considered for crater matching applications, but only considering the model/hardware combination. In fact, we would like to highlight the performance achieved by the MobileNetV3 + SSD model: this architecture with the same hardware (Unibap CPU) manages to decrease latency and obtain a frame rate that exceeds the threshold of 10 frames per second. The model allows the latency to be reduced by -48% and the frame rate to be increased by 53% (for the example case of the FP16 CPU). However, this model is not actually supported by the Myriad board, but considering future developments and projecting the performance boost that the Myriad brings if only the NN model is evaluated, the best performing appears to be the MobileNetV3.

3.5 Future Works

The work reviews and analyzes the importance of the navigation system for ODTS purposes, describing the crater matching technique and its use for the upcoming lunar missions. Specifically a comparison of different NN models and their execution over a space qualified hardware was carried out.

The performances of the network architectures have been measured on two performance parameters: throughput and latency. The results have shown that the MobileNetV2 with SSD is the best solution that satisfies the camera frame rate constraint. The analysis also highlighted that the MobileNetV3 with SSD could be a promising solution in the future if its compatibility issue with the Myriad device will be solved.

This work is seen as a small step in showing how in perspective the use of vision assisted approach can become an interesting hypothesis to reach an autonomous OD system, with the possibility to overcome the still unrealistic (at today) goal of a complete independence from Earth ground infrastructures. This will be a key changing technology in breaking the boundary of the numbers of the nodes for a dedicated navigation lunar infrastructure. Moreover another interesting research field is the possibility to miniaturize the on–board image–based technology to open, in a more realistic way, the use of Small Sat for navigation purpose, which use is affected significantly from the uncertainty of the positioning technique.

Further studies envision the full simulation of the ODTS system, starting from the simulation of the camera acquisition to the final orbit determination.

Part II

Other Works

Chapter 4

Artificial Intelligence–Based Data Path Control in LEO Satellites–Driven Optical Communications

Free Space Optical communication has emerged as a promising technology for high–speed and secure data transmission between ground stations on Earth and orbiting satellites. However, this communication technology suffers from signal attenuation due to atmospheric turbulence and beam alignment precision. Low Earth Orbit satellites play a pivotal role in optical communication due to their low altitude over the Earth surface, which mitigates the atmospheric precipitation effects.

This work introduces a novel data path control law for satellite optical communication exploiting AI–based predictive weather forecasting and a node selection mechanism based on Reinforcement Learning. Extensive simulations on three case studies demonstrate that the proposed control technique achieves remarkable gains in terms of link availability with respect to other state–of–the–art solutions.

This work has been submitted to the International Journal of Satellite Communications and Networking and is available in a pre–print format [96].

4.1 Introduction

In today's ever–connected world, the demand for high–speed, reliable, and secure data transmission has never been greater. The proliferation of data-intensive applications, such as streaming video [97], cloud computing [98], and the Internet of Things (IoT) [99], continues to place unprecedented strain on traditional communication networks. To meet these growing demands, the development of innovative communication technologies has become imperative. Among them, Free Space Optics (FSO) has emerged as a promising solution to address these challenges.

In the telecommunications domain, FSO indicates all those wireless communications which, instead of making use of radio carriers in the form of a radio communication, make use of electromagnetic carriers belonging to the range of optical or infrared frequencies or wavelengths, aimed at transporting information between a transmitter and a receiver [100].

FSO offers several advantages over traditional radio frequency (RF) communication systems when used in space–based applications. In particular, the main benefit of FSO communication relies on data transfer rates. Indeed, a FSO system based on lasers can achieve much higher data transfer rates with respect to RF communications. This is due to the fact that laser light has a much shorter wavelength than RF waves: the wavelength of laser light falls within the optical spectrum, typically in the range of 400 to 700 nanometers (nm), while RF waves can have much longer wavelengths, ranging from millimetres to meters [101]. The shorter wavelength of laser light allows for higher frequency modulation, which means that data can be encoded onto the carrier signal at much higher frequencies. This enables a more significant number of data bits to be transmitted per unit of time.

Moreover, laser communication systems can exploit a larger portion of the electromagnetic spectrum, including multiple wavelength channels, to transmit data simultaneously. This multiplexing capability increases the total data capacity of the communication link.

Other advantages of FSO systems over the RF counterpart rely on (i) the smaller divergence of the laser beam, which enables a higher concentration of optical power [102], (ii) lower interference thanks to a point–to–point communication with a direct line of sight [103], (iii) lower latency over longer distances [104], and (iv) more robust security due to the inherent difficulty to intercept FSO signals without being located precisely in the path of the beam [105].

However, FSO communication systems come also with drawbacks and limitations related to atmospheric turbulence. The latter can significantly impact the performance of FSO system, causing most of the time (i) scintillation of the received optical signal [106], (ii) beam wander [107], and (iii) beam divergence [108].

The FSO link is established by means of an optical ground station (OGS), a facility designed for the reception and transmission of laser signals from and to space assets. In general, the effects of adverse weather conditions on FSO systems become more pronounced as the distance between the transmitter and the receiver increases. This is why a FSO link between an OGS and a satellite is typically operated by means of LEO satellites.

LEO satellites play a pivotal role in the advancement of FSO communication. Situated at altitudes between 180 and 2,000 km above the Earth's surface [109], LEO satellites have relatively short orbital periods, typically completing one orbit around the Earth in 90–120 minutes [110]: this frequent orbiting allows for better coverage and faster data transmission.

Due to their relatively low altitude, atmospheric effects, such as signal attenuation due to rain fade and atmospheric turbulence, have a reduced impact on a FSO system compared to GEO satellites. This results in more reliable and consistent FSO communication links, characterised by low latency, high data throughput and improved signal strength, making them a preferred choice for real–time, high–data–rate FSO applications [111].

Moreover, this type of satellite can be deployed in large constellations [10, 112], which provide continuous global coverage and improve the overall reliability of the FSO communication. This aspect is crucial for applications that require uninterrupted connectivity, such as satellite–based internet services.

LEO satellites are commonly used in the space industry for scientific research and Earth observation purposes and for military operations, as well as for communication, navigation, and remote sensing applications [113].

On the other hand, LEO satellites present two main disadvantages compared with the GEO ones. The first one deals with their shorter lifespan, requiring periodic orbital adjustments to maintain their position in the orbit, or replacements of units within the fleet [114, 115]. The second one is related with visibility issues: since LEO satellites' rotation speed is much higher than the Earth's rotational speed, FSO terrestrial signals have to be handed over to another satellite within the fleet. A satellite handover is performed when the serving satellite is below a minimum elevation angle relative to the corresponding OGS: this may have a significant impact on the communication quality, because of communication loss during the handover process [116].

Despite these downsides, LEO satellites represent the ideal technology for optical satellite communications. However, problems related to laser signal attenuation in the presence of adverse atmospheric conditions remain. To address these technological difficulties, various methodologies have been proposed by the scientific community.

4.2 State of the Art and Original Contribution

Researchers and engineers have created a variety of strategies and technologies to solve the problems of power attenuation in FSO systems due to atmospheric fading. A standard procedure rely on adaptive optics [117, 118]. Systems implementing this technology correct for turbulence–induced distortions by changing the geometry of optical components like mirrors or deformable lenses based on real–time observations of air turbulence. This technique aids in optical beam stabilisation and minimises scintillation effects.

Other techniques rely on filtering and error correction, in which proper filters and modulation methods try to filter out noise coming from the interference of fog or clouds [119]. Since in many situations it is not possible to filter out the noise, it is possible to employ broader laser beams to reduce the effects of beam spreading brought on by turbulence [108]. This strategy, nevertheless, could result in slower data transfer rates [120]. Eventually, another common standard approach is to implement redundant FSO lines, equipping satellites or OGSs with more than one laser communication terminal (LCT) [121], or FSO/RF hybrid systems [122], in order to enhance the communication system reliability.

The aforementioned fading mitigation techniques intervene at the hardware level on the individual receiver or transmitter, but do not take into consideration any changes to the architecture or topology of the communication system.

In order to limit bad weather effects, it is possible to intervene at architecture level linking in a wired fashion two or more OGSs within a same network. In this way, if the signal suffers some degradation in an area, the other OGSs, located in areas where the weather is favourable, may compensate said attenuation. The communication loss is mitigated by continuously forwarding the signal to the OGS(s) under untoward weather conditions, at least until the latter improve. This technique is called site diversity [123].

The site diversity has proven to be a disruptive approach for the reliability of FSO communications, since it (i) enables geographical diversity to reduce the likelihood of simultaneous signal degradation at all sites [123], (ii) realises spatial separation to ensure that the OGSs locations are subject to different weather patterns and atmospheric conditions [124], and (iii) involves using multiple antennas at each site, pointing in different directions or at different elevation angles. This configuration allows the system to quickly switch between antennas to find the clearest signal path, thus improving the link availability and reducing the number of outages or dead times [125].

Although the site diversity technique adds complexity and high cost to the infrastructure, the benefits in terms of improved reliability and availability often justify its implementation, particularly for mission–critical applications.

Said technique employs sophisticated control and switching mechanisms to monitor the quality of signals received at different sites in real–time. When one site experiences signal degradation, the system automatically switches to an alternate site with better signal quality. These switching mechanisms were initially manual and human–driven, while nowadays are usually based on statistical analysis of weather forecasts [126–129], with the switching system being controlled by intelligent algorithms.

The choice of which OGS to point at or to transmit from is driven by a series of Key Performance Indicators (KPIs) and follows an optimal routing/resource allocation logic [130, 131]. The most important KPI for any satellite communication system is the link availability, but other design drivers for the multi–station site diversity algorithms may include (i) the energy consumption for the movement/re-pointing of LCTs, which impacts on the total power budget for the on–board payload, (ii) the topology of the OGSs network (i.e., specific OGSs network topologies may prevent the possibility to re–route the user traffic from one OGS to the others), (iii) user plane latency and jitter, and (iv) on-board switching capabilities.

Since the installation of redundant OGSs may represent a waste of investment for the network operator, several works focused on the minimisation of the number of required OGSs to guarantee a minimum given system performance [132–134]. A different optimisation approach relies on the hypothesis that OGSs have been already positioned and the problem focuses on how to choose the set of OGSs to connect to in order to maximise the availability. In [135] authors calculate the correlated and uncorrelated availability for OGS networks in the scenario of space–to–ground optical communication links with GEO satellites. An efficient optimisation algorithm is presented, in order to choose the best OGS starting from five years of cloud data. It is shown how many OGSs deployed in a very wide area can guarantee a network availability near to 100%. A complementary optimisation approach is proposed in [136], with the selection of the minimum number of ground stations satisfying the monthly availability requirements of the total network, so minimising service and maintenance costs. Eventually, in the scenario presented in [137], the optimisation process consists in selecting the best ground station among several candidates, trying to provide a reliable connectivity through large–scale site diversity. Results show that the optimal choice mostly depends on the altitude and the zenith angle of the set of ground stations.

Recent advancements in AI, particularly in the field of RL, have opened up new possibilities for optimising satellite communication strategies. Authors in [121] propose an AI–based predictive handover strategy for optical communications between a

GEO satellite equipped with two LCTs and an OGS network, making use of machine learning–based weather forecasts. Other works making use of AI and RL focused on the resource allocation and traffic splitting for RF satellite communications [138–141], shifting attention from the OGS network level to the one of the LEO constellation.

However, none of the above–mentioned works have tackled the issue of defining an intelligent handover and path planning procedure for a FSO–based point–to–point communication system between terrestrial OGS networks and a LEO fleet.

In this work a mixed Deep Learning (DL) for weather forecasts and RL approach for intelligent signal routing is proposed, in order to tackle the problem of minimising the outage probability. The main innovations of the present work are:

- multi–hop data routing between two OGS networks that cannot communicate directly, but only passing through a LEO satellite fleet;
- weather predictions over the OGSs areas via Supervised Learning exploiting historical hourly weather data;
- a centralised control law realised through an intelligent agent exploiting the RL framework with an intrinsic optimisation of the link availability.

The remainder of this chapter is organised as follows: Section 4.3 provides general background on time series machine learning and the RL foundations. Section 4.4 presents the mathematical modelling of LEO satellites' and Earth's dynamical motion. In Section 4.5 extensive simulations show the effectiveness of the proposed approach with respect to other benchmark solutions, and, eventually, Section 4.6 sums up the carried out research, pointing out the achieved results, describing as well its limitations and outlining future research directions.

4.3 Preliminaries

In this section the mathematical foundations of RNNs and RL control will be presented.

4.3.1 LSTM Neural Networks

A RNN is a type of artificial neural network designed for processing time sequences of data. Unlike traditional feedforward neural networks, which have a fixed architecture, RNNs are equipped with loops or recurrent connections that allow them to store memory about previous inputs [142]. This memory enables RNNs to process sequences of data, such as time series, natural language, or any other data with a temporal or sequential structure.

The LSTM network was first suggested in [143] to address the well–known problem of vanishing gradient that characterises RNNs. The LSTM structure is therefore ideally adapted to handle time-series data, such as the one we are addressing in our work. By specifying a certain time window $\mathcal{T}_p$ of length T_W, the AI model tries to predict weather at time $k+1$ by looking at the actual weather encountered in the T_W previous time instants, i.e.:

$$\mathcal{T}_p = \Big\{k, k-1, \dots, k-T_W-1\Big\}. \tag{4.1}$$

At their core, LSTMs are comprised of memory cells that enable them to store and manipulate information over extended sequences. These memory cells have three crucial components:

1. Cell State: it is like a conveyor belt that runs through the entire LSTM network. It can transport information across time steps without much modification. The cell state can be updated, allowing it to capture relevant information and discard irrelevant details;

2. Hidden State: also known as the output state, carries information from previous time steps to the current one. It acts as a working memory that helps LSTMs remember past information that is crucial for making predictions or decisions;

3. Gates: LSTMs employ three types of gates to control the flow of information:

   - Forget Gate: this gate decides what information from the cell state should be discarded or kept. It takes as input the previous hidden state and the current input and outputs a value between 0 and 1 for each component of the cell state, where 0 means "forget" and 1 means "keep";
   - Input Gate: this gate determines what new information should be added to the cell state. It computes a candidate cell state and decides which parts of it should be added to the current cell state;
   - Output Gate: the output gate controls what information should be output as the hidden state. It takes the current cell state and the input, and it generates the new hidden state.

In this work, each LSTM network is trained on local OGS meteorological data, because if not so it would be difficult for a single predictor to generalise across the various climates of the OGSs' geographic regions, which can actually be located at very different latitudes.

4.3.2 Markov Decision Process and Reinforcement Learning

Reinforcement Learning is one of the branches of machine learning: it deals with intelligent agents performing actions over an environment and then observing its state and the reward function [144]. The agents' goal is to find a policy which maximizes the expected cumulative reward, without being aware of the dynamical equations governing the environment (data–driven control). Usually, a RL problem is formalized through a Markov Decision Process (MDP), defined through a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where:

- $\mathcal{S}$ is the state space;
- $\mathcal{A}$ is the action space;
- $P(s'|s, a)$ is the probability that the environment transitions from state s to state s' when the agent chooses the action a;
- $R(s, a, s')$ is the immediate reward the agent gets when transitioning from state s to s' after taking action a;
- $\gamma \in [0,\ 1)$ is the discount factor, representing the agent's preference for immediate rewards ($\gamma \approx 0$) over the future ones ($\gamma \approx 1$).

The objective of the agent in an MDP is to find a policy $\pi : S \to A$ that maximizes the cumulative function of the rewards over a (potentially) infinite horizon [144]. Usually, said function is defined as:

$$G_k = \sum_{i=0}^{\infty} \gamma^k R_{k+i+1}. \tag{4.2}$$

When the transition probability $P(\cdot)$ is not known, it is possible to rely on RL techniques, in which the agent learns the optimal policy through the experience. All MDPs handled through RL require the estimation of a value function or an action–value function $Q_\pi(\cdot)$ for a given policy $\pi(\cdot)$ [144]:

$$Q_\pi(s, a) = \mathbb{E}_\pi[G_k | s_k = s, a_k = a], \forall s \in \mathcal{S}, \forall a \in \mathcal{A}. \tag{4.3}$$

Action–value functions satisfy recursive relationships through the Bellman Equation, which expresses a link between the action–value function of a state and the

Algorithm 1 Q-Learning.

```
    Inputs: learning rate $\alpha \in [0, 1)$; discount rate $\gamma \in [0; 1]$; small $\varepsilon > 0$
    Output: $Q(s, a)$
 1: Initialize $Q(s, a), \quad \forall s, \quad \forall a$
 2: for all episodes do
 3:     reset $s$
 4:     for each step of the episode do
 5:         choose action $a$ following $\varepsilon$–greedy policy
 6:         perform action $a$ and observe $s'$, $r'$
 7:         perform Q–Learning update rule over $Q(s, a)$
 8:         $s \leftarrow s'$
 9:     end for
10: end for
```

action–value function of the next state:

$$\begin{aligned} Q_\pi(s, a) &= \mathbb{E}_\pi[G_k | s_k = s, a_k = a] \\ &= \mathbb{E}_\pi[R_{k+1} + \gamma G_{k+1} | s_k = s, a_k = a] \\ &= \sum_{s'} P(s'|s, a)(R + \gamma \sum_{a'} \pi(a'|s') Q_\pi(s', a')) \end{aligned} \tag{4.4}$$

where (s', a') is the next state–action couple with respect to (s, a). Hence, solving an MDP through RL means finding the optimal action–value function $Q^*(s, a) = \max_\pi Q_\pi(s, a)$ for which it holds the Bellman principle of optimality [144]:

$$Q^*(s, a) = \sum_{s'} P(s'|s, a)(R + \gamma \max_{a'} Q^*(s', a')). \tag{4.5}$$

One of the most popular and used RL algorithm to estimate $Q^*(s, a)$ is the Q–Learning [145]; said algorithm updates the Q–values according to the following law:

$$Q(s_k, a_k) \leftarrow (1 - \alpha) Q(s_k, a_k) + \alpha[R_{k+1} + \gamma \max_a Q(s_{k+1}, a)] \tag{4.6}$$

where α is the so–called learning rate. A popular choice for the policy $\pi(\cdot)$ is the ε–greedy policy through which the agent selects with probability ε a random action and with probability $1 - \varepsilon$ the action associated to the maximum value in the Q table. For finite MDPs, it has been proven that the Q–Learning algorithm is able to converge to the optimal Q–function if the Q–Learning update rule given by (4.6) is used and if the learning rate α satisfies the following conditions:

$$\sum_k \alpha_k = \infty, \quad \sum_k \alpha_k^2 < \infty. \tag{4.7}$$

This condition requires that $\alpha \in [0; 1)$ which translates in the fact that each state-action pair is visited infinitely often. By adopting an ε–greedy policy, this condition can be stochastically satisfied. The pseudocode of the algorithm has been reported in Algorithm 1.

4.4 Modelling

Let us consider a FSO–like communication system made by two OGS networks, one transmitting data from N_{tr} OGSs and the other one acting as receiver with N_{re} stations. Each of the two zones can be subject to different atmospheric conditions, going from sunny to cloudy to stormy, which affect the data delivery from the transmitting to the receiving zone. The two sets of OGSs cannot communicate using terrestrial wired or wireless technologies, but they must rely on a LEO constellation composed of N_{sat} satellites. The communication is a point–to–point one realised through laser beams. The system scenario is depicted in Figure 4.1.

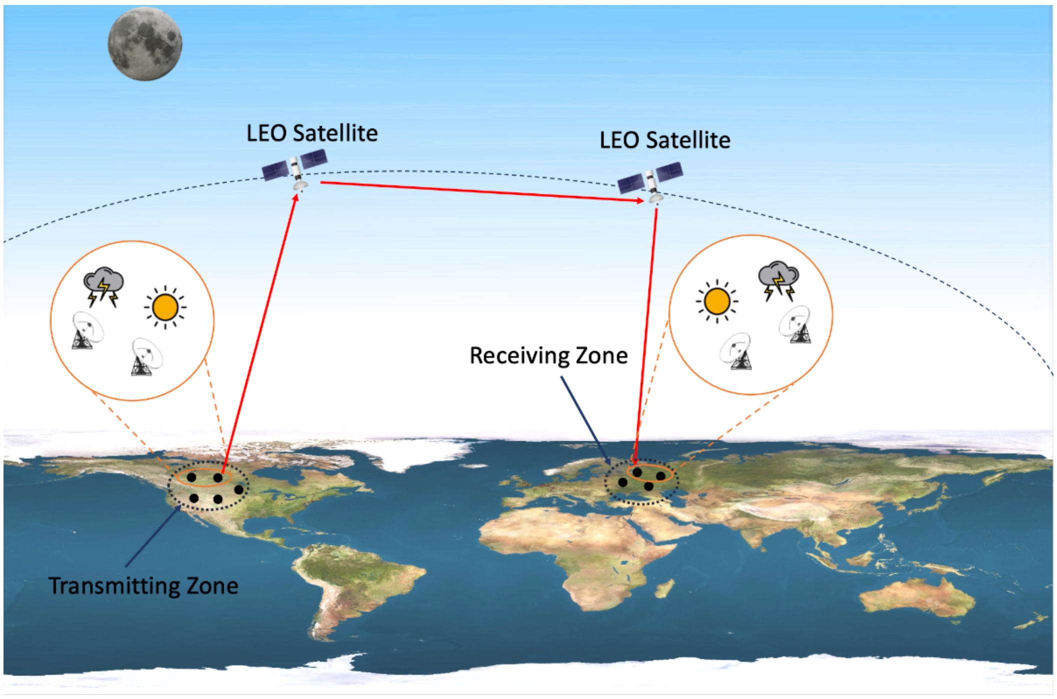


Figure 4.1. System Scenario.

In what follows, a detailed mathematical modelling of the overall communication system is presented, including the formulation of the orbiting LEO satellites equations of motion, the ground–to–satellite and inter–satellite visibility assessment, and the MDP characterisation.

4.4.1 Satellite Equations of Motion

Low Earth Orbit satellites are considered one of the best options for satellite communication due to their short orbital period, which provides wide coverage and an high service availability.

In order to define a LEO constellation of satellites, the orbit itself must be characterised. Given an inertial frame of reference and an arbitrary epoch (a specified point in time), exactly six parameters are necessary to unambiguously define an arbitrary and unperturbed orbit. These are the semi–major axis a, the eccentricity e, the inclination i, the argument of perigee ω, the longitude of the ascending node Ω, also denoted as RAAN for geocentric orbits, and the true anomaly f [12, 29].

The orbital parameters can be used to compute, at every epoch, the position and velocity of the satellite around that orbit. To describe the motion of satellites, it is usually used a coordinate frame which is inertial and fixed with respect to the stars, namely the ECI reference frame [146]. In particular the x–y plane coincides with the equatorial plane of Earth. The x–axis is permanently fixed in a direction relative to the celestial sphere, which does not rotate as Earth does. The z–axis lies at a 90° angle to the equatorial plane and extends through the North Pole (see Figure 4.2, taken from `https://en.wikipedia.org/wiki/Earth-centered_inertial`).

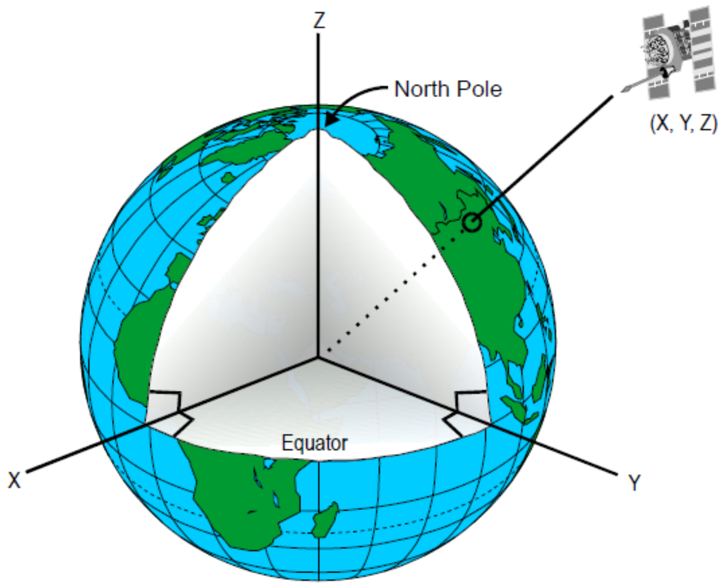


Figure 4.2. Earth Centered Inertial reference frame.

Let us define as $R_x(\phi)$, $R_y(\eta)$ and $R_z(\psi)$ the standard rotation matrices:

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \tag{4.8}$$

$$R_y(\eta) = \begin{bmatrix} \cos\eta & 0 & -\sin\eta \\ 0 & 1 & 0 \\ \sin\eta & 0 & \cos\eta \end{bmatrix} \tag{4.9}$$

$$R_z(\psi) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.10}$$

Algorithm 2 shows how to pass from the orbital parameters to the satellite position and velocity in the ECI coordinates.

Algorithm 2 Orbital Parameters to ECI coordinates.

```
**Inputs**: $a, e, \Omega, i, \omega, f$
**Parameters**: $\mu = 3.986004418 \times 10^{14} [m^3/s^2]$
**Outputs**: r, v

$p = a(1 - e^2)$                              ▷ semilatus rectum
$cf = \cos f, sf = \sin f$
$r = p/(1 + e(cf))$                           ▷ Safe Division
$v = \sqrt{mi/p}$                             ▷ Safe sqrt and safe division

Define a rotation matrix based on angles and axes
ang $= \begin{bmatrix} \omega & i & \Omega \end{bmatrix}^T$
axes $= \begin{bmatrix} 3 & 1 & 3 \end{bmatrix}^T$
$M = \text{ang2mat}(\text{ang}, \text{axes})$

Compute position and velocity in ECI
Transpose $M$
$r \leftarrow rM \begin{bmatrix} cf & sf & 0 \end{bmatrix}^T$
$v \leftarrow vM \begin{bmatrix} -sf & e + cf & 0 \end{bmatrix}^T$
```

At this point it is possible to define the satellite equations of motion as a second–order differential equation which is dependent on the satellite position vector r:

$$\ddot{r} = -\mu \frac{r}{\|r\|^3} \tag{4.11}$$

where $\|r\|$ is the euclidean norm of the position vector and $\mu = 3.986004418 \times 10^{14}$

[$\mathrm{m}^3/\mathrm{s}^2$] is the geocentric gravitational constant.

4.4.2 Visibility Analysis

In order for the satellite to exchange information with an OGS or with another satellite there is a condition which needs to be analysed, the visibility. The latter is a very important concept since it can determine if a certain information exchange can happen or not and how good is the communication channel in terms of noises. Visibility can be of two kind: (i) geometric visibility, which is related to the fact that the relative position vector between one satellite and the other does not have to intersect the Earth, and (ii) electronic visibility, which deals with analysing the elevation angle and Carrier to Noise ratio (C/N_0). The angle of elevation is the angle between the horizontal line and the line of sight which is usually above the horizontal line. The C/N_0 expresses how high is the noise component with respect to the information carrier: the lower the ratio is, the more the noise is prevalent and vice–versa.

Since this work does not focus on the quality of the communication link, some assumptions have been made to simplify the analysis:

1. the information exchanged between the satellite and the OGS and between one satellite and another one is always good with a negligible amount of noise;

2. the satellite is visible by the OGS if the elevation angle is greater than a certain threshold, in order to exclude the case of interference of buildings in the vicinity of the OGS;

3. the satellite is visible with respect to another one if the geometric visibility condition is satisfied.

In the following, the implementation of the visibility algorithms related to assumptions 2 and 3 will be detailed.

4.4.3 Ground to Satellite Visibility

As already explained, a satellite is considered visible from an OGS if the elevation angle is above a certain threshold. The elevation angle is computed with respect to the horizontal plane of the OGS, so the East–North–Up (ENU) coordinate frame has been considered, which is the reference frame of the ground station's antenna. This implies a change of coordinates of the satellite position and velocity vectors from the ECI reference frame to the ENU frame. This transformation can be performed by applying two rotations starting from the original coordinates: the first one to pass from the ECI to the ECEF coordinates, the second one to pass from the ECEF to ENU coordinates.

Since the ECEF reference frame is non–inertial and is rotating along with the Earth, a new dynamic equation must be introduced to take this rotation into account. Defining θ as the angle of rotation of the Earth, the latter's rotational dynamics can be easily written as:

$$\dot{\theta} = \omega_E \tag{4.12}$$

where $\omega_E = 2\pi/86400 \approx 7.29 \times 10^{-5}$ [rad/s] is the angular velocity of the Earth. In Algorithms 3 and 4 the steps to compute the two rotations are detailed. In the following, the notation x_{RF} with RF $\in$ {ECI, ECEF, ENU} denotes the reference frame of the generic vector x, while the notation $x_{\mathrm{RF},c}$ with $c \in \{x, y, z\}$ denotes the three components of the generic vector x expressed in the RF coordinates.

Algorithm 3 ECI to ECEF coordinates transformation.

```
**Inputs**: $r_{\mathrm{ECI}}, v_{\mathrm{ECI}}, \theta$
**Parameters**: $\omega_E$
**Outputs**: $r_{\mathrm{ECEF}}, v_{\mathrm{ECEF}}$

$R = R_z(\theta)$
$r_{\mathrm{ECEF}} = R r_{\mathrm{ECI}}$
$a = v_{\mathrm{ECI},x} + \omega_E r_{\mathrm{ECI},y}$
$b = v_{\mathrm{ECI},y} - \omega_E r_{\mathrm{ECI},x}$
$c = v_{\mathrm{ECI},z}$
$\tilde{v} = \begin{bmatrix} a & b & c \end{bmatrix}^T$
$v_{\mathrm{ECEF}} = R\tilde{v}$
```

Algorithm 4 ECEF to ENU coordinates transformation.

```
**Inputs**: $r_{\mathrm{ECEF}}, \phi, \nu$                    ▷ $\phi$, $\nu$: lat and long of the GS
**Outputs**: $r_{\mathrm{ENU}}$

$R = \begin{bmatrix} -\sin\nu & \cos\nu & 0 \\ -\cos\nu\sin\phi & -\sin\nu\sin\phi & \cos\phi \\ \cos\nu\cos\phi & \sin\nu\cos\phi & \sin\phi \end{bmatrix}$
$r_{\mathrm{ENU}} = R r_{\mathrm{ECEF}}$
```

As a last step, from the ENU coordinates it is possible to compute the Azimuth (A), Elevation (E) and Range (ρ) of the satellite with respect to the OGS's antenna. For our case only the elevation angle will be used in the visibility analysis. Algorithm 5 details the mathematical steps to compute these three parameters.

Algorithm 5 ENU to Azimuth, Elevation, Range parameters.

```
Inputs: $r_{\mathrm{ENU}}$
Outputs: $A, E, \rho$

$\rho = \|r_{\mathrm{ENU}}\|$
$\sigma = r_{\mathrm{ENU}}/\rho$
$E = \arcsin \sigma_z$ [rad]
$A = \arctan(\sigma_x, \sigma_y)$ [rad]
```

4.4.4 Satellite to Satellite Visibility

Due to the short field of view of the LEO satellites, in order to exchange information between two sites far away from each other, a constellation of satellites is needed. This implies the creation of a communication link between two satellites of the same constellation in order to reach the remote site efficiently. The concept of visibility applies also in this case. To simplify the analysis only the geometric visibility is considered. Algorithm 6 details the procedure for the geometric visibility check.

Algorithm 6 Geometric Visibility Check between satellite A and B.

```
Inputs: $r_A$, $r_B$
Parameters: $R_{\mathrm{Earth}} = 6378136.3$ [m]
Outputs: isSatVis [bool]

Initialize output
isSatVis = False
norm = $\|r_A\|$

if $r_A == r_B$ then                                   ▷ Same point in space
    isSatVis = True
    return isSatVis
else
    $r_C = r_A - r_B$                                  ▷ Relative position vector
    min dist = Minimum distance between $r_C$ and the centre of the Earth
    if min dist $\geq R_{\mathrm{Earth}}$ then
        isSatVis = True
    end if
    return isSatVis
end if
```

4.4.5 Markov Decision Process Formulation

The system dynamics described above has been translated to a MDP in order to exploit the RL framework. In Figure 4.3 the proposed RL algorithm workflow is detailed.

The state space is

$$\mathcal{S} =< t >, \quad t = 0, \ldots, T \tag{4.13}$$

where t is the generic time step and T is the final step within the transmission window period.

The action space is

$$\mathcal{A} =< \mathrm{OGS_T}, \mathrm{SAT_1}, \mathrm{SAT_2}, \mathrm{OGS_R} > \tag{4.14}$$

where $\mathrm{OGS_T}$ is the index of the transmitting OGS, $\mathrm{SAT_1}$ is the index of the first satellite receving data from the transmitter, $\mathrm{SAT_2}$ is the index of the second satellite receiving data from the first one, and $\mathrm{OGS_R}$ is the index of the receiver.

Eventually, the reward function models the success rate of the end–to–end handover and is defined in the following way:

$$R = \begin{cases} +1, & \text{if transmission is successful} \\ -1, & \text{otherwise} \end{cases} . \tag{4.15}$$

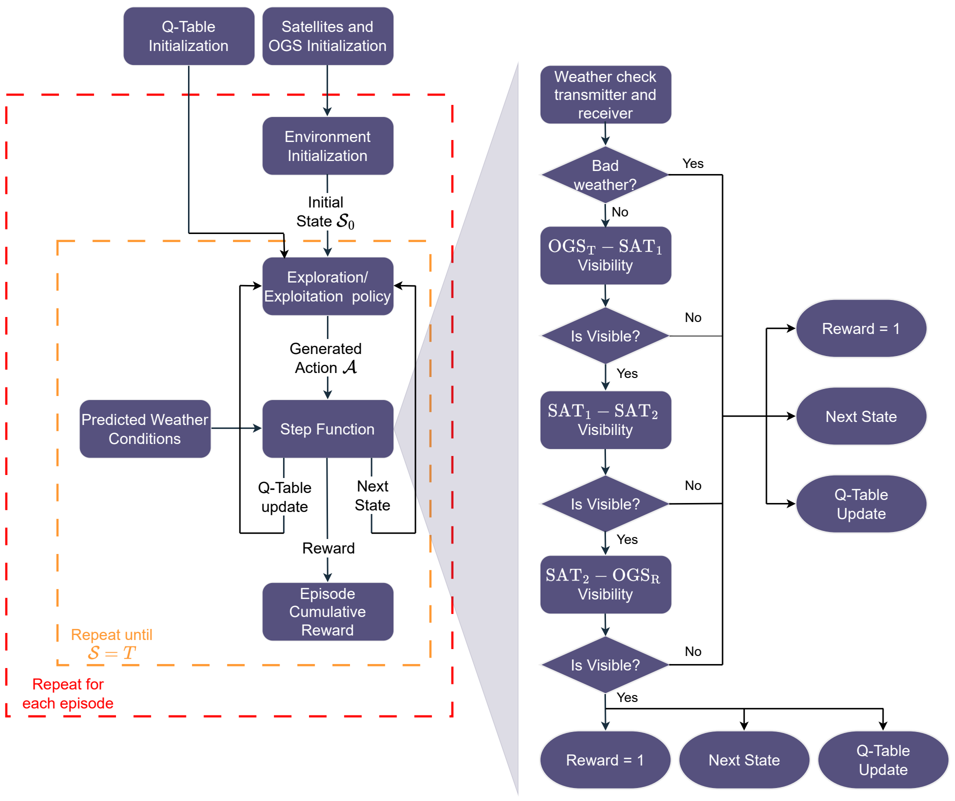

Q-Table Initialization
Satellites and OGS Initialization
Environment Initialization
Initial State $\mathcal{S}_0$
Exploration/ Exploitation policy
Generated Action $\mathcal{A}$
Predicted Weather Conditions
Step Function
Q-Table update
Next State
Reward
Episode Cumulative Reward
Repeat until $\mathcal{S} = T$
Repeat for each episode
Weather check transmitter and receiver
Bad weather?
Yes
No
$\mathrm{OGS_T} - \mathrm{SAT_1}$ Visibility
Is Visible?
No
Yes
$\mathrm{SAT_1} - \mathrm{SAT_2}$ Visibility
Is Visible?
No
Yes
$\mathrm{SAT_2} - \mathrm{OGS_R}$ Visibility
Is Visible?
No
Yes
Reward = 1
Next State
Q-Table Update
Reward = 1
Next State
Q-Table Update

Figure 4.3. Reinforcement Learning algorithm workflow.

4.5 Simulations and results

In order to simulate and validate our control approach, two geographical areas from two different continents have been considered, namely:

1. the east coast of United States and Canada, with $N_{\text{tr}} = 10$ transmitting OGSs located in the main cities, as in Figure 4.4;

2. the territory of Israel, with $N_{\text{re}} = 6$ receiving OGSs, shown in Figure 4.5.

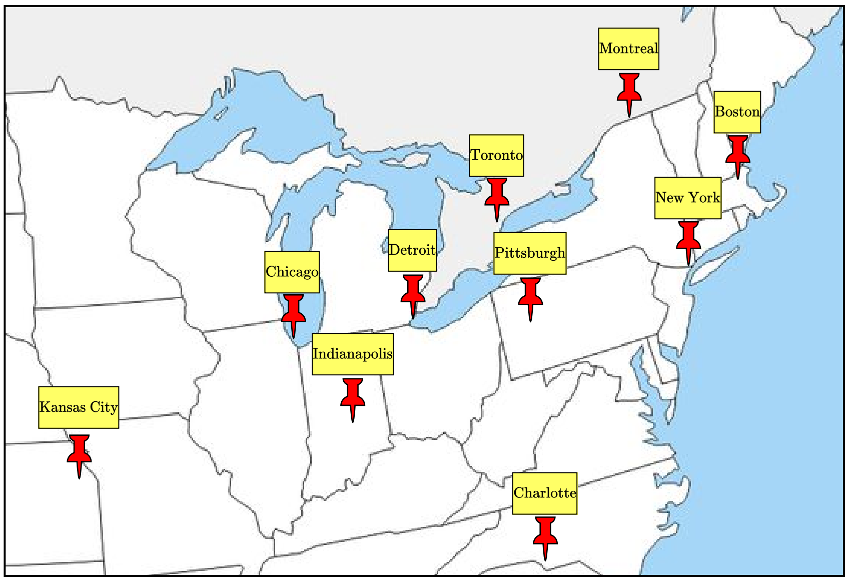


Figure 4.4. Map of the transmitting OGSs in the east coast of North–America.

To perform the weather forecast for all the OGS zones, the LSTM deep neural network was trained on a publicly available weather dataset [147] covering approximately 5 years of weather data (from October 1, 2012 to November 30, 2017), with temporal resolution $T_R = 1\,\text{h}$.

The available features for training are the following:

- humidity;
- atmospheric pressure;
- wind direction;
- temperature;
- wind speed;
- month of the year;

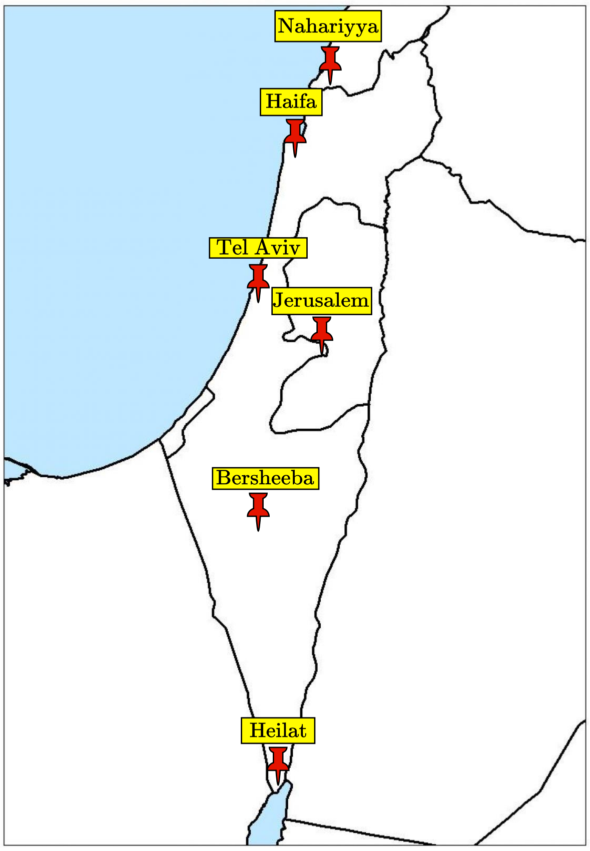


Figure 4.5. Map of the receiving OGSs in Israel.

- weather conditions within the time window prediction $\mathcal{T}_p$.

The missing data for each feature was filled in by taking up the numerical value of the feature of the previous entry: this approach makes it possible not to break the hourly time sequence of the meteorological data.

As per the weather, the dataset contains a very detailed description of the weather conditions. The latter have been mapped into binary labels for training the model: label 0 has been assigned to clear sky, few/scattered clouds and haze, which correspond to mild weather conditions allowing satellite–OGS communication, whereas the label 1 indicates inclement weather which does not allow a successful data transmission.

For the training phase, the data from October 1, 2012 up to December 20, 2016 have been selected. The model accuracy has been evaluated by splitting the remaining part of the dataset with respect to the four seasons, in accordance with the 2016 and 2017 astronomical tables [148].

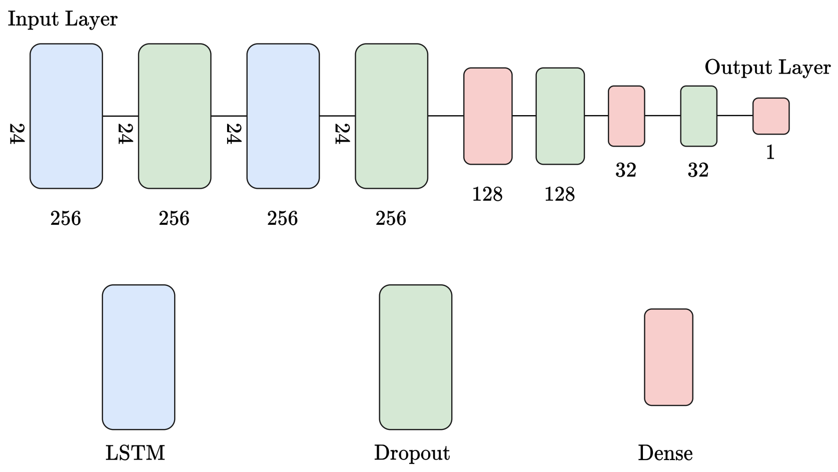


Figure 4.6. LSTM Neural Network Architecture. The network is made by two LSTM layers and two Dense layers, each one followed by a Dropout layer, with the final output layer having one neuron representing bad weather probability.

The chosen LSTM model architecture is depicted in Figure 4.6 and the selected hyperparameters are the following:

- number of epochs $E = 5$;
- Adam optimizer with constant learning rate $\eta = 0.001$;
- time window length $T_W = 24$;
- dropout rate $\zeta = 0.2$.

The LSTM model performance on unseen data (from December 21, 2016 to November 30, 2017) against all seasons and per each city, in terms of test accuracy, have been reported in Tab. 4.1. It is evident that the neural network model is able to predict correctly almost all the weather conditions within the test set, thus representing a powerful tool to estimate in advance the precipitation or thick clouds probability over the zone in which the OGS is located.

Table 4.1. LSTM accuracies per season.

City	Winter	Spring	Summer	Autumn
New York	0.984	0.986	0.982	0.989
Montreal	0.988	0.981	0.963	0.974
Boston	0.993	0.989	0.987	0.989
Chicago	0.973	0.978	0.952	0.971
Charlotte	0.976	0.976	0.972	0.975
Pittsburgh	0.991	0.984	0.979	0.988
Detroit	0.997	1.000	0.996	0.998
Kansas City	0.992	0.990	0.983	0.990
Toronto	0.999	0.999	0.985	0.988
Indianapolis	0.992	0.986	0.981	0.992
Beersheba	0.944	0.956	0.966	0.998
Tel Aviv District	0.977	0.955	0.966	0.991
Eilat	0.932	0.923	0.968	0.999
Haifa	0.944	0.982	0.987	0.999
Nahariyya	0.989	0.985	0.945	0.997
Jerusalem	0.991	0.981	0.959	0.998

The RL–based controller hyperparameters for the training phase have been selected as follows:

- $\gamma = 0.9$;
- $\varepsilon_0 = 1$ with episodic decay law with respect to the generic episode η:

$$\varepsilon(\eta) = \mathrm{e}^{-\dfrac{\eta}{\beta N_{\mathrm{ep}}}}$$

  with $\beta = 0.2$ being the decay rate and N_{ep} the number of episodes;

- $\alpha_0 = 1$ with episodic decay law with respect to the generic episode η:

$$\alpha(\eta) = \mathrm{e}^{-\dfrac{\eta}{1000}}.$$

As for the evaluation phase, the controller performance has been figured out over a transmission period of $T = 2$ days.

The AI–based control law has been evaluated in terms of link availability, defined as follows:

$$L_A = \frac{N_S}{N_T} \tag{4.16}$$

where N_S is the number of times a successful data transmission is achieved, and N_T is the total number of transmissions attempted.

Results with respect to the above–defined KPI have been compared with other benchmark routing approaches in the FSO domain, listed as follows:

- B1. Both transmitting and receiving OGSs and both LEO satellites are chosen randomly;
- B2. Transmitting and receiving OGS are chosen with a reactive approach based on the current weather condition, and the satellites are chosen with the min range technique, following the reasoning and modelling provided in [149];
- B3. Transmitting and receiving OGS are chosen with a reactive approach based on the current weather condition, and the satellites are chosen as those with the maximum elevation angle;
- B4. Transmitting and receiving OGS are chosen with the LSTM–based weather forecasts, and the satellites are chosen with the min range technique;
- B5. Transmitting and receiving OGS are chosen with the LSTM–based weather forecasts, and the satellites are chosen as those with the maximum elevation.

The proposed control approach has been tested over three different case studies, in which the communication between the two OGSs networks is realised with different LEO constellations:

- Case study 1. $N_{\text{sat}} = 15$ satellites from the Iridium constellation;
- Case study 2. $N_{\text{sat}} = 15$ satellites from the Starlink constellation;
- Case study 3. $N_{\text{sat}} = 30$ satellites given by the combination of satellites from case study 1 and case study 2.

The satellite orbital parameters and generic data have been gathered via Two–Line Elements (TLEs) files from [25]. A Two–Line Element file is a data format encoding a list of orbital elements of an Earth–orbiting object for a given point in time.

The propagation of the satellites motion over time is performed by using the 4–th order Runge-Kutta algorithm as integrator with fixed time step $dt = 1$ minute.

All the simulations have been carried out using Tensorflow framework on Python3.10 on a machine equipped with an Intel Core i5–10210U CPU and 16GB RAM.

4.5.1 Iridium Constellation

In this case study the number of episodes for training the RL controller has been set as $N_{\text{ep}} = 100$. The season–related cumulative reward trend over the training episodes is shown in Figure 4.7. In all the four cases, the reward converges to a

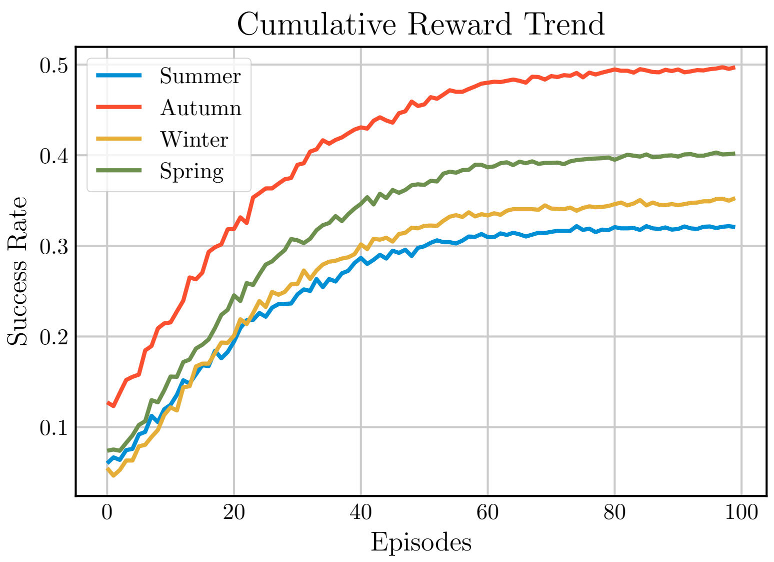


Figure 4.7. Season–related reward trend of the RL controller for the Iridium case study.

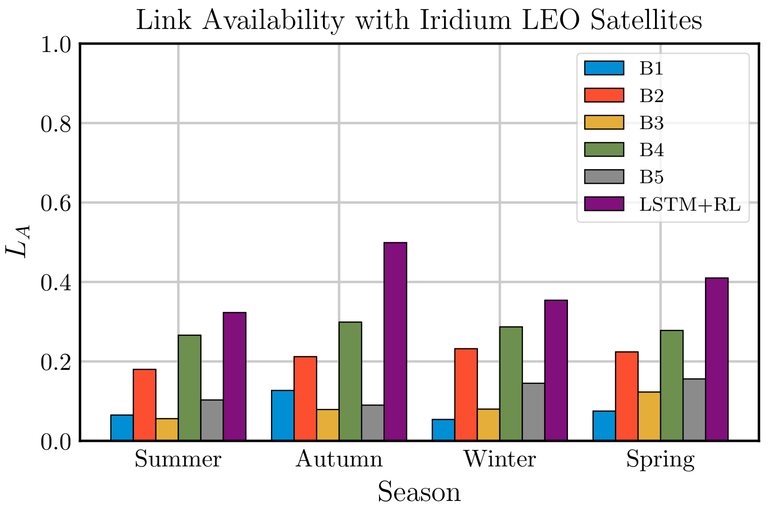


Figure 4.8. Season–related link availability comparison for the Iridium case study.

steady–state value in terms of data transmission success rate, which is higher in the autumn season due to the presence of a higher number of hours with favourable weather conditions both at transmitting and receiving zone. The comparison of the performance of the proposed approach with respect to the benchmark solutions is shown in Figure 4.8. It is worth noting that the RL controller together with a LSTM–based weather prediction achieves higher link availability with respect to the other standard techniques for FSO communication.

4.5.2 Starlink Constellation

In this case study the number of episodes for training the RL controller has been set as $N_{\text{ep}} = 100$. The cumulative reward trend is shown in Figure 4.9. For all seasons, the reward converges to a steady–state value in terms of data transmission success rate, as in the previous case study with the Iridium constellation.

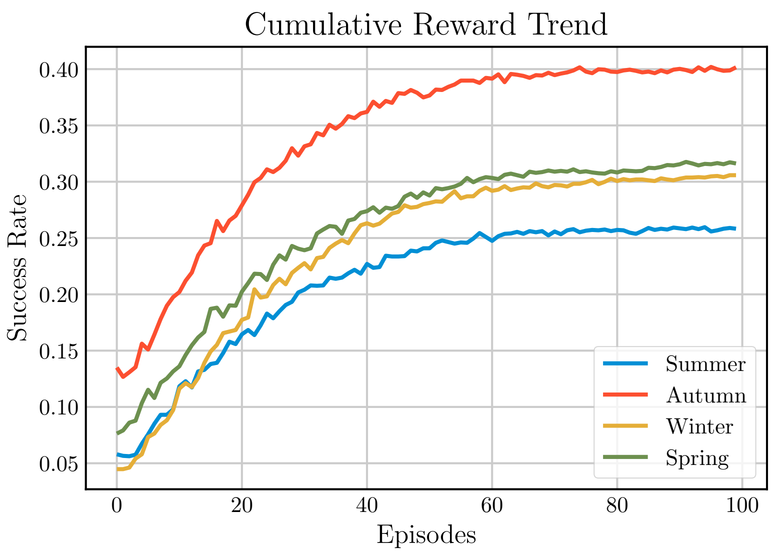


Figure 4.9. Season–related reward trend of the RL controller for the Starlink case study.

Figure 4.10 depicts the comparison of the performance of the RL controller with respect to the benchmark solutions. The proposed control strategy achieves the best performances in terms of link availability. However, it shall be noticed that the overall performances are worse with respect to those achieved by means of the Iridium constellation. As an example, in the autumn season the RL controller guarantees $L_A = 0.401$ using Starlink satellites and $L_A = 0.499$ with Iridium satellites: similar results hold for the other seasons. This is due to the fact that the Starlink constellation orbits have been designed to cover mainly the North–American continent, thus guaranteeing poor coverage within the Israel territory.

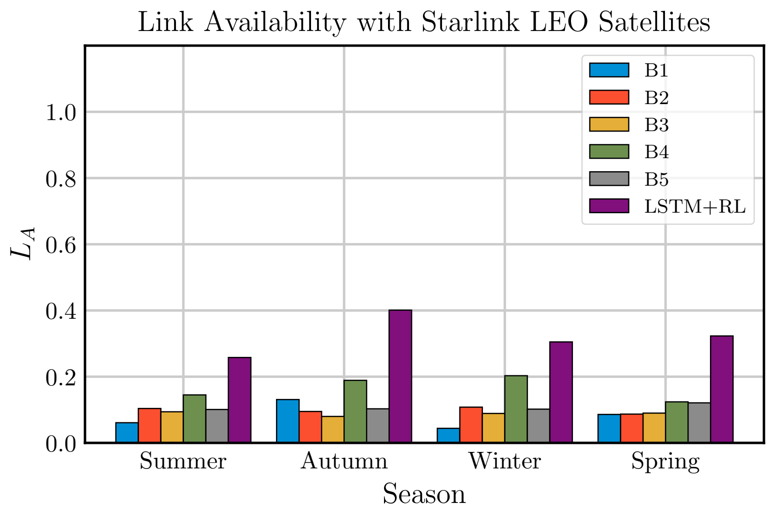


Figure 4.10. Season–related link availability comparison for the Starlink case study.

4.5.3 Mixed Constellation

In the last case study the number of episodes for training the RL controller has been increased to $N_{\text{ep}} = 300$, in order to allow a broader exploration due to the availability of double the amount of LEO satellites with respect to the previous case studies.

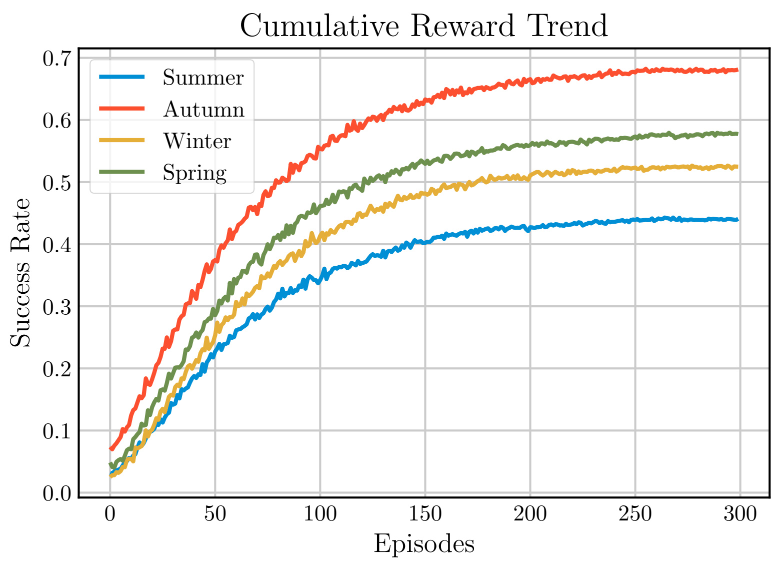


Figure 4.11. Season–related reward trend of the RL controller for the mixed case study.

Figures 4.11 and 4.12 show training and evaluation performances against all seasons. As expected, the increased number of satellites guarantees higher cumulative reward trend and, hence, link availability for the FSO transmission. This is due to

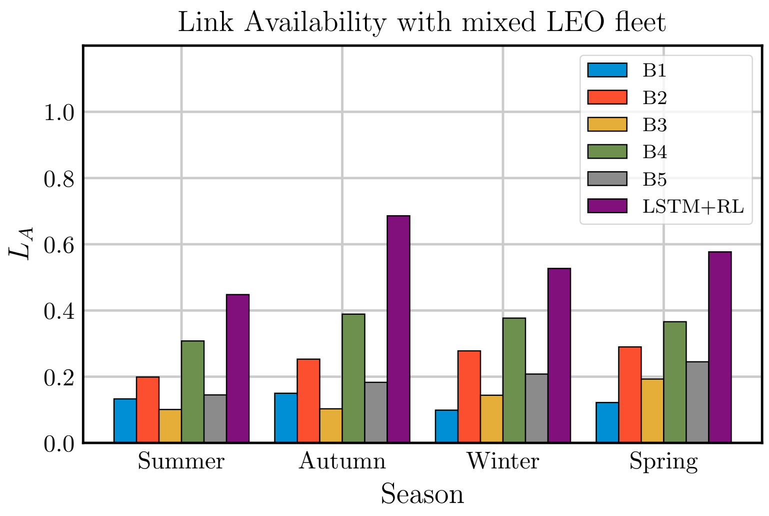


Figure 4.12. Season–related link availability comparison for the mixed case study.

the fact that increasing the number of satellites leads to a wider coverage over the Earth surface: this allows to establish a successful communication with guaranteed inter–satellite visibility for longer periods. Also in this case, the performances of the proposed control algorithm are better than the benchmark ones.

4.6 Conclusions and Future Works

In this work a mixed AI and RL approach for FSO point–to–point communication has been proposed. This technique exploits weather prediction algorithms to improve the quality of the communication link, as well as a dynamical data–driven optimisation for maximising the link–availability in a data transmission scenario between two terrestrial OGS networks communicating through LEO satellites. The proposed decision and control approach has been compared with several benchmark solutions, achieving better performances in all seasons over the three analysed case studies in which different LEO constellations have been exploited.

However, some limitations hold. The developed RL–based algorithm does not take care about the frequent rotation of the LCT due to the OGS–satellite and inter–satellite dynamical switching, and no physical considerations on signal attenuation and beam spreading due to atmospheric condition and relative distance have been made.

Future works could focus on the problems defined above, proposing control algorithms that take signal attenuation into consideration, introducing link budget and beam spreading modelling, also with strategies aimed at saving energy on the various LEO devices.

Chapter 5

PhiFireAI

The *PhiFireAI* project has been developed in collaboration with the Advanced Technologies research group of Thales Alenia Space Italy as an answer to an ESA challenge on the development of an AI algorithm for Earth Observation. The name of the challenge was “AI4EO – OrbitalAI Challenge”.

Our solution has been selected as part of the top 5 projects, from a pool of over 70 participants, which will proceed to the second phase of the challenge, called Incubation Phase. In this phase ESA experts will be supporting the teams in order to finetune and improve the algorithm with a chance to be selected as one of the two winners of the competition. The top two teams will have their application deployed on the Φsat–2 satellite.

The rest of the chapter is organized in the following way: in Section 5.1 an overview of the Φsat–2 mission is presented along with a description of the on–board processor that will manage all the AI applications; in Section 5.2 the solution will be presented and its usefulness to the wider community will be explained; in Section 5.3 the proposed solution will be detailed; Section 5.4 will present the obtained results: the model performances on the validation metrics and the testing results on the Unibap iX5–100 processor will be shown and discussed; at last in Section 5.5 the carried out research, the achieved results and its limitations will be summed up, outlining possible future research directions.

5.1 Introduction

The European Space Agency has an exciting vision: creating a thriving ecosystem of EO applications using edge computing in space. This groundbreaking challenge addresses the current obstacles in the flow of Earth observation data and paves the way for the next generation of applications. By employing advanced AI techniques to process data directly onboard spacecraft, we can unlock the immense potential of onboard intelligence for EO. This approach promises enhanced efficiency, agility, autonomy, and reconfigurability in EO. On–board processing in satellites is revolutionising space data by enabling faster and more efficient data transmission, reducing the dependence on ground–based processing, and enabling real–time decision–making. In EO, end users require valuable insights and optimal decisions with minimal delay. That's why significant research efforts have been dedicated to exploring onboard intelligence for EO applications, such as early detection of natural disasters, vessel incidents, and gas leaks. On–board intelligence empowers us to identify low–quality EO data, such as cloud–covered satellite images or remote sensing images with limited relevant information and discard them. This not only saves costs but also minimizes the need for data transmission to Earth.

5.1.1 Φsat–2 Mission Overview

As an initiative to promote the development and implementation of innovative technologies on–board Earth Observation missions, in 2018 ESA kicked off the first Φsat–related activities with the aim to enhance the already ongoing FSSCAT project with AI. Thanks to the success of the Φsat–1 experiment, it was decided to promote the Φsat line to full missions and this time under the name of Φsat–2. The call has been issued at the end of 2019 seeking new ideas and innovative approaches to on–board EO data processing and AI exploitation and after the evaluation phase, the project implementation started at the end of 2020. The mission is now due for launch in Q4 2023 / Q1 2024.

The selected Φsat–2 concept will provide a combination of on–board processing capabilities (including AI) and a medium to high resolution Visible to Near Infra–Red (VIS/NIR) multispectral instrument able to acquire 8 bands (7 + Panchromatic). These resources will be made available to a series of dedicated applications (hereafter called App) that will run on–board the spacecraft that have been partially pre–selected during the initial phases of the mission design (4 out of 6 to be tested in the operative phase). The remaining 2 Apps will be selected via a global competition open to the AI and EO communities.

The Φsat–2 spacecraft is composed of:

- The payload chain, consisting of:
  - Multiscape 100 Optical Imager supplied by Simera Innovate GmbH (CH), that acquires images of the tasked area, performs radiometric correction and first order, open–loop band co–alignment of the acquired images. It offers 4.75m resolution, 19.4km swath width in 7 multispectral bands (plus a panchromatic band) at the reference altitude of 500km;

  - Primary On–Board Computer (OBC), that combines functionalities of the payload and platform computer. As part of the payload chain, the primary on–board computer will host the SDK (so called NanoSat MO Framework) with AI Apps;
  - Secondary OBC, that is interfaces with the primary on–board computer and runs the image pre–processing algorithms;
  - CogniSat AI processor supplied by Ubotica Technologies Ltd (IE), that hosts AI model and performs inference.

- Spacecraft platform responsible for providing all necessary services: host and command the payload as required by the Mission Operations and the Payload Operations Centres and download the payload and housekeeping data to the ground stations.

According to the baseline launch service and to ensure the required Sun illumination conditions, the parameters of the reference orbit used for the Φsat–2 design purposes are reported in Table 5.1.

Launch Date	Q4 2023/Q1 2024
Baseline Orbit Altitude	500 km
LTDN	11:00 PM
Inclination	97.404 deg (SSO)
Eccentricity	0
RAAN	-111.75 deg

Table 5.1. Φsat–2 Orbital Parameters.

The spacecraft is a 6U CubeSat based on the standard Open Cosmos OpenSat 6U platform. Φsat–2 spacecraft in deployed and stowed configurations are shown in Figures 5.1 and 5.2, respectively.

Its reference frame is defined with respect to the orbit:

- +**X**: velocity vector;
- +**Y**: Orbital Angular Momentum;
- +**Z**: Zenith.

The Φsat–2 spacecraft is designed for a 14–month lifetime from launch with potential extension up to 2 more years. Figure 5.3 shows the nominal top–level timeline of the mission, and Table 5.2 Summary of Mission PhasesTable 4 provides a description of the activities undertaken during each phase.

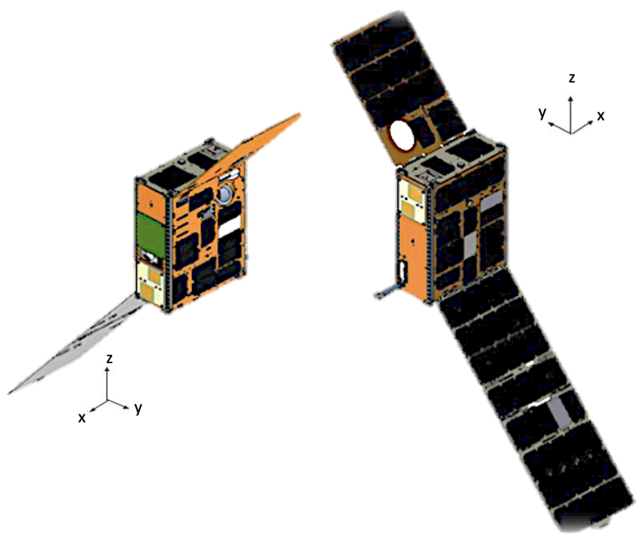


Figure 5.1. Φsat–2 deployed configuration.

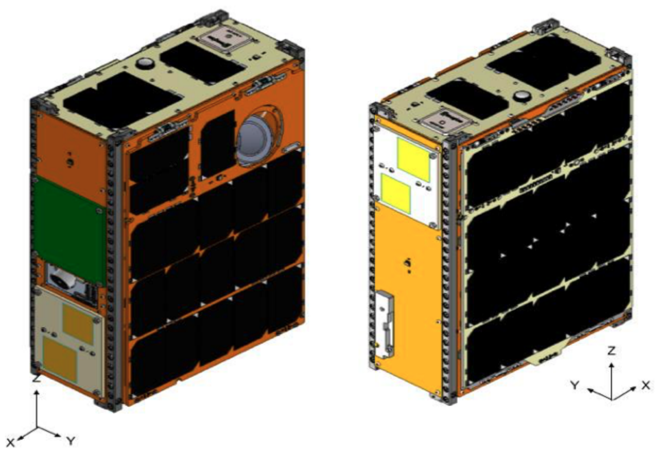


Figure 5.2. Φsat–2 stowed configuration.

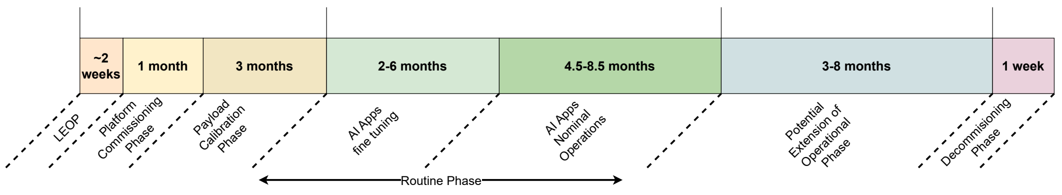

~2 weeks
1 month
3 months
2-6 months
4.5-8.5 months
3-8 months
1 week
LEOP
Platform Commissioning Phase
Payload Calibration Phase
AI Apps fine tuning
AI Apps Nominal Operations
Potential Extension of Operational Phase
Decommisioning Phase
Routine Phase

Figure 5.3. Φsat–2 Mission Timeline.

Phase	**Duration**	**Activities**
Launch and Early Operation Phase (LEOP)	2 weeks	• Launch and separation; • First acquisition signal; • Satellite detumbled and stabilised attitude achieved; • Orbital knowledge and determination; • Initial health checks of the platform.
Platform Commissioning Phase	1 month	• Platform & Payload commissioning.
Payload Calibration Phase	3 months	• Payload Calibration; • On–board L1B image processing chain commissioning.
Routine Phase	10.5 months	• Fine tuning of AI apps; • Uplink of two additional AI apps; • Nominal spacecraft operations; • Spacecraft maintenance.
Potential Extension of the Operational Phase	3–8 months	• Depending on the spacecraft orbital decay and spacecraft health status nominal operations can be extended for another 3–8 months.
Decommissioning Phase	1 week	• Acquisition of end–of–life attitude; • Spacecraft decommissioning.

Table 5.2. Summary of Φsat–2 Mission Phases.

5.1.2 Φsat–2 Payload

MultiScape100 instrument from Simera Innovate GmbH (CH) is a push–broom imager. The imager provides continuous line–scan imaging in 8 spectral bands in the visible and near–infrared (VNIR) spectral range. A push–broom instrument obtains the images by scanning along the ground track as the spacecraft is orbiting the Earth (see Figure 5.4 below). The scan is done for each spectral band separately.

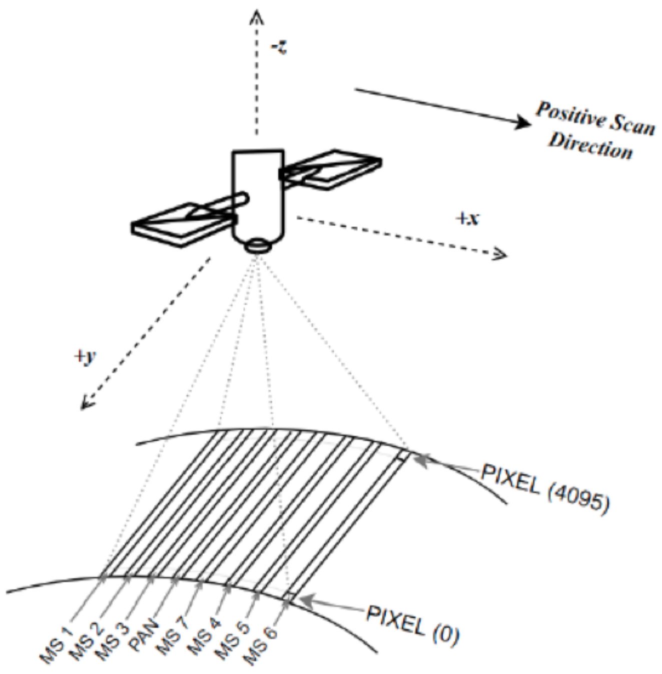


Figure 5.4. Φsat–2 Imager Ground Projection.

In Table 5.3 below information on each spectral band including its line number on detector plane is provided.

Band	Centre Wavelength (nm)	FWHM Bandwith (nm)	HPP Cut–On (nm)	HPP Cut–Off (nm)
#0: PAN	625	250	500.0	750.0
#1: MS 1	490	65	457.5	522.5
#2: MS 2	560	35	542.5	577.5
#3: MS 3	665	30	650.0	680.0
#4: MS 4	705	15	697.5	712.5
#5: MS 5	740	15	732.5	747.5
#6: MS 6	783	20	773.0	793.0
#7: MS 7	842	115	784.5	899.5

Table 5.3. Φsat–2 spectral bands.

The ground sampling distance (GSD) for each band from 500 km orbit is 4.75 m. The modulation transfer function (MTF) for separate spectral bands is expected to be between 3.9% and 7.2% at Nyquist Frequency. The SNR is expected to be between

54 and 129 for the multispectral bands. The expected SNR for the PAN band is 256. The images of each spectral band are retrieved by the on–board computer and stored in its internal memory. The images are then radiometrically corrected, the spectral bands are co–registered, and pixels are Geo–located. The data is then ready to be used by AI applications. By default, the roll angle will be set to 0. The spacecraft is capable of capturing images with at least 15 degrees roll angle while keeping pitch at 0 degrees. No pitch manoeuvres are planned to be used during imaging. Unless specified by the end user, the spacecraft will only capture images when the Solar Zenith Angle is between 0 and 90 degrees, i.e., when the Earth's surface is illuminated by the sun.

5.1.3 CogniSat AI processor

The Ubotica CogniSAT–XE1TM CubeSat Board (Figure 5.5) brings the power of Edge CV and AI compute acceleration to a PC/104 form–factor for SmallSat and CubeSat missions. It is built around the Intel® MovidiusTM MyriadTM 2 CV and AI COTS VPU whose 12 vector cores provide high–performance parallel and hardware accelerated compute within a low power envelope. CogniSatTM combines the power efficient compute of the Myriad 2 VPU with a wide range of interfaces and peripherals, providing broad flexibility for integration into satellite platforms. Either Gigabit Ethernet or USB2.0/3.0 can be used as the primary control and data interface to the board, enabling data rates sufficient to handle many CV and AI applications at near–streaming throughput. Common NN frameworks (e.g., TensorFlow, PyTorch, Caffe) can be used for NN model development and training, with the model subsequently imported into Intel's OpenVINOTM toolkit for targeting to the Myriad device. CogniSatTM leverages the broad range of pre–qualified models and layers available within OpenVINOTM. Custom CV pipelines can easily be deployed and executed on CogniSatTM using the CVAI ToolkitTM software toolkit. The software supports application–specific CV and ISP pipelines that utilise a combination of the power–efficient Myriad 2 streaming hardware blocks and software filters implemented on the vector engines. Deployment to the hardware platform involves the transfer of only a single configuration file, and runtime updates enable the updating of pipelines without requiring application re–compile or system reboot.

5.1.4 Nanosat MO Framework

The NanoSat MO Framework (NMF) is a software framework for CubeSats based on CCSDS Mission Operations services. It facilitates not only the monitoring and control of the nanosatellite software applications, but also the interaction with the

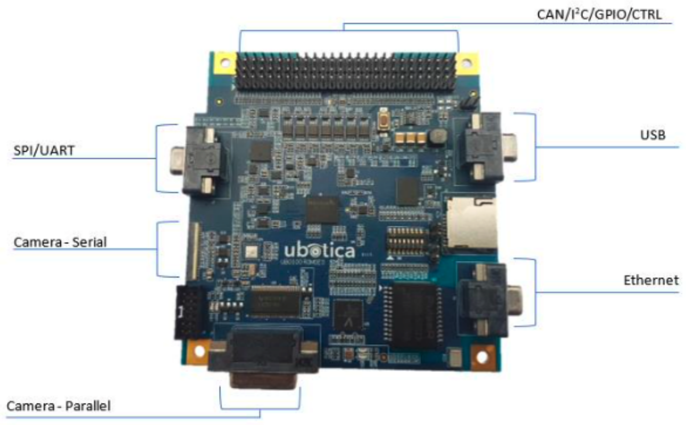


Figure 5.5. Ubotica's CogniSatTM Platform.

nanosatellite platform. This is achieved by using the latest CCSDS standards for monitoring and control, and by exposing services for common peripherals that are available in nanosatellite platforms, such as, GPS, Camera, ADCS, and others. Furthermore, it is capable of managing the software on–board by exposing a set of services for software management. In simple terms, the NanoSat MO Framework introduces the concept of apps in space that can be installed, and then simply started and stopped from ground. Apps can retrieve data from the nanosatellite platform through a set of well–defined Platform services. Additionally, the framework includes CCSDS standardised services for monitoring and control of apps. An NMF App can be easily developed, distributed, and deployed on a spacecraft. Just like Android and iOS, ESA made a software framework for flight and ground software of CubeSats which allows flight software to become Apps that run in space. The main objective of the NanoSat MO Framework is to facilitate the development of software for small satellites and to simplify its orchestration. For example, new software can be easily deployed in a satellite just by starting and stopping Apps.

The core functionalities of the framework are:

- Monitoring and control of AI applications;
- Easy access to platform peripherals via services;
- Simple on–board software management: deploy and run AI applications;
- Updating AI applications and AI models;
- Deleting AI applications and AI models.

In the Φsat–2 Mission the NMF will provide the possibility to integrate models developed by the AI experts, wrapping the AI models and enabling them to:

- Deploy a model on the CogniSat AI processor;
- Retrieve image from the camera;
- Apply the model in prediction on the acquired image.

The NanoSat MO Framework allows the AI App developers to make their App using well–defined interfaces and APIs that allow the App to reach the on–board devices via a set of services. The framework by itself will also prevent conflict when different Apps will compete to access a resource (e.g., imager) which in any case will be scheduled via the Payload Orchestrator Center. NanoSat MO Framework and AI applications will operate in a dedicated memory space allocated specifically for them, which will not intersect with the rest of the software running on the OBC. In case of NanoSat MO Framework and any of the AI apps malfunction, OBC flight software will not be affected.

5.2 Solution and Contribution to the Community

The *PhiFireAI* solution aims to enhance the detection of wildfires using image–based satellite data and to provide onboard early–warning information, suggesting mitigation actions or targets that could be used to prevent the spread of the wildfire. As Earth observation satellites could help firefighting efforts detect wildfires directly from space, we would like to provide helpful information to intervene in case of wildfire promptly, supplying intelligence for incident commanders, firefighters and civil protection (see Figure 5.6).

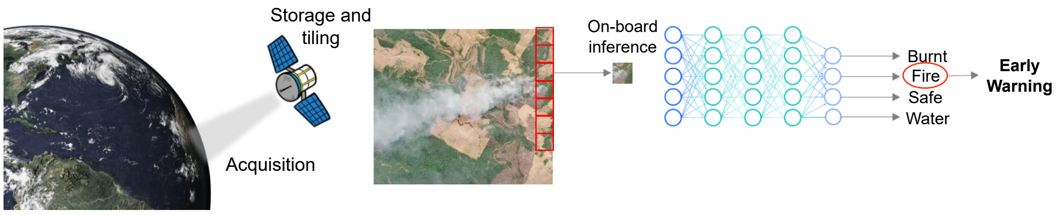


Figure 5.6. PhiFireAI solution scheme.

Due to the importance that detecting fire contributes to the ongoing study of climate change and people's safety, there have been several investments to obtain fire maps worldwide. However, the most efficient models use disparate data sources, such as image–capturing technology from satellites, aircraft, drones, artificial intelligence and cloud computing, to predict how new fires will burn (based on drought monitoring) and, therefore, prevent future mega–fires. These models are crucial for prevention as the climate crisis worsens yearly. Therefore, this work aims not only to ascertain the presence of wildfires but also to define their characteristics in terms of dangerousness. Additionally, it seeks to apprise authorities of surrounding land attributes and available water sources crucial for firefighting. The ultimate outcome could involve seamless integration with other data sources, thereby enhancing the quality of the end–user product.

The solution is able to tackle this problem by addressing a classification task employing a model designed as a classifier, equipped with four outputs to predict the ultimate class for each individual tile (Safe–0, Fire–1, Burnt–2, Water–3). To explore the influence of various input band combinations on performance, we maintain the same model structure while altering the input layer dimensions. After executing inferences across all tiles comprising a complete acquisition, we generate a comprehensive overview of the entire image.

The innovative aspect lies in the capacity of run the entire algorithm on a space qualified hardware and the ability to give in real time important information to the final user. Our solution can be used to add information to current fire monitoring databases but also, autonomously, is able to detect an hazard directly onboard the satellite. In the future, the count of final classes could also be expanded to provide a greater volume of real–time information.

5.3 Methods

In this section a detailed description of the proposed solution is presented. A comprehensive overview of the dataset preparation process, offering detailed insights into the sequential steps involved is given. Following this, the neural network architecture is presented, accompanied by a thorough discussion of the training and validation strategies, with a specific emphasis on the adopted validation metrics.

All the solution–related code has been written in a Python environment. For the training script, TensorFlow 1.15.5 library has been used due to a hard constraint imposed by ESA for the OrbitalAI challenge.

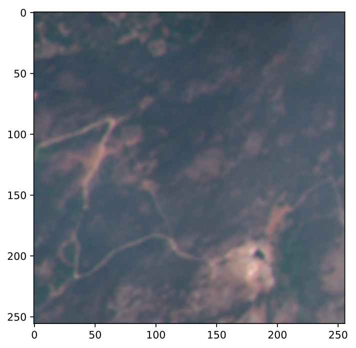
5.3.1 Data Preparation

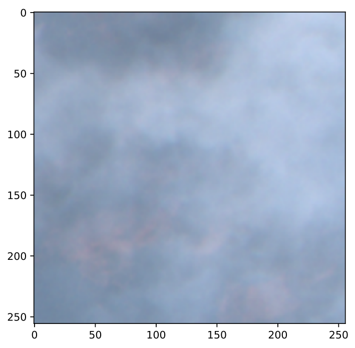
For the training, validation and testing of this work, a dataset comprising acquisitions of several of Europe's largest fires over the past five years was generated ad–hoc.

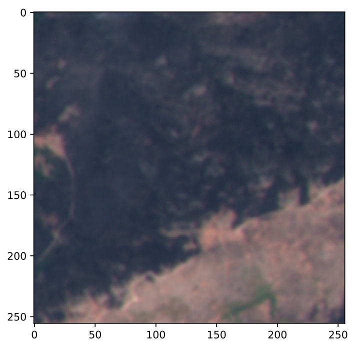
The public database of historical wildfires of European Forest Fire Information System (EFFIS) was used to derive information on the Geo–reference shape, approximate date and size of the largest fires in Europe reported between 2018 and 2022. [150–154].

At first, Sentinel–2 L1C images were automatically downloaded using the Sentinel

Hub API using the centroid of the fire shape and taking all acquisitions 7 days before and 7 days after the approximate date in the EFFIS database. A simulator, provided in the frame of OrbitalAI challenge, was used to derive Φsat–2 bands starting from Sentinel–2 L1C images. The simulated image, of dimension $4096 \times 4096 \times 8$ was divided into $256 \times 256 \times 8$–pixel tiles and a simple graphical user interface was implemented to simplify the task of manually labelling each tile into its respective class: Safe, Fire, Burnt and Water. Figure 5.7 shows an example of the labelled images.


(a) *Input image for class 0–Safe.*


(b) *Input image for class 1–Fire.*


(c) *Input image for class 2–Burnt.*

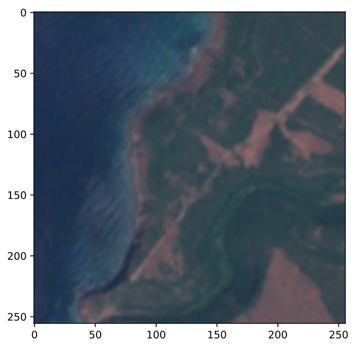

(d) *Input image for class 3–Water.*

Figure 5.7. Example of input images with the corresponding labels.

The dataset was then divided into train and validation with an 80/20 ratio, and two new acquisitions were generated separately for the test set. The obtained training set was unbalanced: the number of images related to the Safe class was way higher than the other classes. In order to solve this problem, a data augmentation procedure has been implemented, balancing the number of samples of the other classes. In particular, horizontal and vertical flipping, along with a ninety–degree rotation, were applied to augment the training set. Upon completion of the dataset preparation, our training set comprised a total of more than twenty–five thousand tiles, while the validation set consisted of over three thousand tiles. These statistics highlight the comprehensiveness of our dataset, laying the foundation for robust wildfire classification model training.

5.3.2 Neural Network Architecture

The main scope of the solution is to detect and classify correctly the wildfires given a full image acquisition by the Φsat–2 satellite. Moreover, this process has to be performed on–board the satellite, introducing a constraint on the size of the neural network weights based on the satellite on–board storage capabilities.

To approach the classification problem, a classifier neural network architecture has been developed from scratch. In order to give the relevant authorities (mainly firefighters) more information on how to properly intervene in the presence of wildfires, additional features have been introduced in the form of classification labels. In particular, the characterization of burnt, safe and water areas has been considered a useful input for the relevant authorities jointly with the detected wildfire areas. This leads to a multi–class classification problem with four classes.

Table 5.4. Neural Network Specifications

Input	Operator	*#out*	NL
$256^2 \times C$	Conv2d	16	ReLU
$256^2 \times 16$	Conv2d	16	ReLU
$256^2 \times 16$	MaxPool 2×2	-	-
$128^2 \times 32$	Conv2d	32	ReLU
$128^2 \times 32$	Conv2d	32	ReLU
$128^2 \times 32$	MaxPool 2×2	-	-
$64^2 \times 64$	Conv2d	64	ReLU
$64^2 \times 64$	Conv2d	64	ReLU
$64^2 \times 64$	MaxPool 2×2	-	-
$32^2 \times 128$	Conv2d	128	ReLU
$32^2 \times 128$	Conv2d	128	ReLU
$32^2 \times 128$	MaxPool 2×2	-	-
$16^2 \times 128$	Dropout	-	-
$16^2 \times 128$	Flatten	32768	-
32768	Dense	784	ReLU
784	Dense	128	ReLU
128	Dense	64	ReLU
64	Dense	4	ReLU

At the same time, to cope with the on–board constraints, the neural network architecture has been kept as simple as possible. It is composed by a set of eight convolutional layers followed by a set of four fully connected layers, also called dense layers. In addition, max pooling layers [155] are added every two convolutional layers and a flattening layer is used to transition from the convolutional to the dense layers. Dropout strategy has been used in order to avoid overfitting [156]. Softmax has been chosen as the activation function [157]. The neural network architecture specifications are shown in Table 5.4.

In the table C represents the number of channels of the input image. The size of the model is 310 Mb.

5.3.3 Training and Validation Strategy

In order to find the best candidate models, several combinations of input channels/bands have been adopted. Since the only variable was the number of channels of the input image, only the number of channels of the first layer of the network has been changed over several training procedures. The structure of the model as well as the hyperparameters have all been mantained. The training hyperparameters used are reported in Table 5.5.

Hyperparameter	**Description**
Train batch size	128
Validation batch size	32
Optimizer	adam
Loss Function	SparseCategoricalCrossEntropy
Epochs	40
Early Stopping	Yes. Patience = 5

Table 5.5. Training Hyperparameters.

The validation of the model is a two–step process. The first validation has been performed by cross–validation after every training epoch. The batch size for the validation step is reported in Table 5.5 and the metric for this step was simply the accuracy curve.

After training, another validation step has been performed on the models. In this case several metrics have been used. Below a list of them with a brief description:

- *Confusion Matrix*: the confusion matrix provides a clear and detailed summary of the model's predictions and their accuracy by comparing them to the actual ground truth. It consists of four essential components: True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN);

- *Precision*: precision is the number of true positives divided by the number of total positive predictions. It is referred to the proportion of correct predictions among all predictions for a particular class;

- *Recall*: true positive divided by the true positive and false negative. Recall measures the model's ability to predict the positives. It is referred to the proportion of examples of a specific class that have been predicted by the model as belonging to that class;
- *F1–score*: harmonic mean of precision and recall;
- *Support*: the number of samples of the true response that lie in that class;
- *ROC and AUC*: ROC is a probability curve and AUC represents the degree or measure of separability. It tells how well the model is classifying each class. Higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1. The ROC curve is plotted with True Positive Rate (TPR) against the False Positive Rate (FPR) where TPR is on the y–axis and FPR is on the x–axis. In the case of multiclass classification, a notion of TPR or FPR is obtained only after binarizing the output and exploiting the One–vs–Rest scheme, which compares each class against all the others (assumed as one).

A comprehensive discussion on the obtained results is presented in the next section.

5.4 Results and Discussion

In this section the training and validation results are detailed and discussed along with the further steps adopted in order to choose the final model. In addition the inference pipeline used to test the model is presented along with its performances on the Unibap iX5-100 space–qualified processor.

5.4.1 Training and Validation Results

After multiple trainings, three combinations of bands (see Table 5.3 for the list of spectral bands) resulted in having the best performances:

- The combination of bands B1, B2 and B3;
- The combination of bands B1, B2, B3 and B0;
- The combination of bands B3, B2, B1.

For simplicity they will be named *model–1*, *model–2* and *model–3* from now on, respectively. This result is further confirmed during the post–training validation phase. In Tables 5.6–5.8 the performance reports of the three models are reported. It can be seen how all the performances are above 90% considering the single classes and all the classes together (average values).

	precision	**recall**	**f1–score**	**support**
class 0	0.98	0.98	0.98	2291
class 1	0.93	0.97	0.95	397
class 2	0.94	0.93	0.94	421
class 3	0.97	0.97	0.97	217
accuracy			0.97	3326
macro avg	0.96	0.96	0.96	3326
weighted avg	0.97	0.97	0.97	3326

Table 5.6. Metrics classification report for model–1.

	precision	**recall**	**f1–score**	**support**
class 0	0.99	0.98	0.98	2291
class 1	0.92	0.96	0.94	397
class 2	0.93	0.96	0.95	421
class 3	1.00	0.98	0.99	217
accuracy			0.97	3326
macro avg	0.96	0.97	0.97	3326
weighted avg	0.98	0.97	0.97	3326

Table 5.7. Metrics classification report for model–2.

	precision	**recall**	**f1–score**	**support**
class 0	0.98	0.98	0.98	2291
class 1	0.96	0.94	0.95	397
class 2	0.91	0.96	0.93	421
class 3	0.96	0.99	0.97	217
accuracy			0.97	3326
macro avg	0.95	0.97	0.96	3326
weighted avg	0.97	0.97	0.97	3326

Table 5.8. Metrics classification report for model–3.

In Figures 5.8–5.10 are shown the plots of the ROC curves and the associated AUC values in the *One–vs–Rest* scheme. It can be noticed how the value of AUC is always greater than 95% for all classes. This means that the models have at least a 95% probability of classifying correctly an input image.

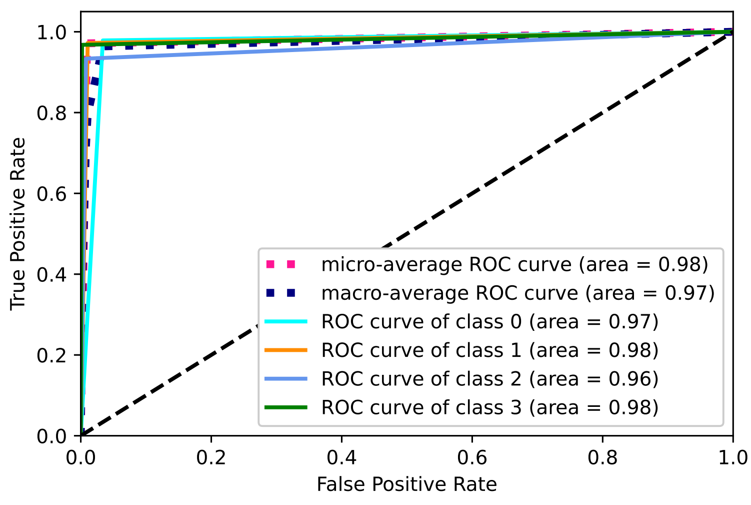


Figure 5.8. ROC curve for model–1 in One–vs–Rest scheme.

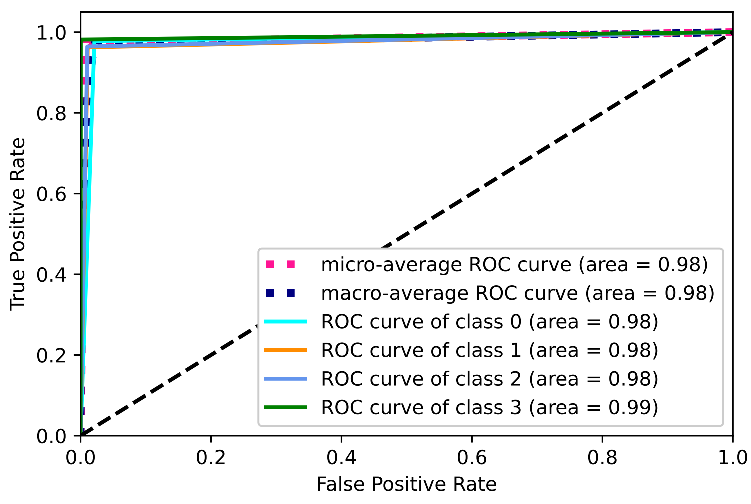


Figure 5.9. ROC curve for model–2 in One–vs–Rest scheme.

This validation analysis has shown that all the selected models are valid candidates to be used for the classification task. In order to choose the best one among them, a different criterion has been adopted. This criterion is linked to the analysis of the confusion matrices of the models with respect to the Fire class. Of the four parameters characterizing them, the ones which have been taken into consideration are the FP and FN. In particular, the best model has been chosen by emphasizing a low number of FN related to the Fire–1 class. Since the aim of this project was to be as accurate as possible in the detection of wildfires, an high value of FN means that

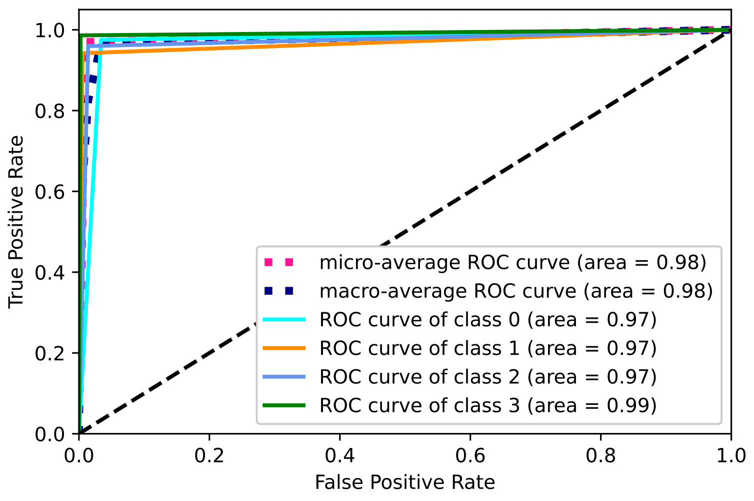


Figure 5.10. ROC curve for model–3 in One–vs–Rest scheme.

the model has high probability of classifying a wildfire area as one of the other three classes. So this parameter should be as low as possible. In Table 5.9 are reported the values of FN and FP of the three models.

Model	False Negatives	False Positives
model–1	2.77%	7.43%
model–2	3.78%	7.73%
model–3	5.79%	4.10%

Table 5.9. False Negatives and False Positives values of the three models for the Fire class.

The values are expressed as percentage of the total number of Fire–1 class labels. As it can be seen, the first model is the one which is considered the best based on the previous assumption. With this choice we accepted the fact of having an higher False Positives rate (w.r.t. to the third model for example).

The chosen model has been quantized in order to reduce the computational load and it has been scaled to uint8 bit precision. The obtained model size is 24.4 Mb. This conversion may lead to a loss of performances. In order to check if this is the case, the quantized model has been subject to another round of validation by using the same metrics as before. In Table 5.10 are shown the performance of the quantized model over the same validation dataset. It can be seen how the performances are almost the same as the original model, which means that we have almost no loss of performances.

This means that, with a reduction of more than 90% of computational power, the

	precision	recall	f1–score	support
class 0	0.99	0.98	0.98	2291
class 1	0.91	0.97	0.94	397
class 2	0.94	0.94	0.94	421
class 3	0.98	0.95	0.97	217
accuracy			0.97	3326
macro avg	0.95	0.96	0.96	3326
weighted avg	0.97	0.97	0.97	3326

Table 5.10. Metrics classification report for model–1 after quantization and scaling.

quantized model offers very similar performances of the original one. Considering the processing capabilities of modern EO satellites, this solution can be effectively deployed and executed on–board in parallel to other applications (AI–based or not) without impacting the overall system in terms of computational resources.

5.4.2 Inference Pipeline

In order to test the chosen model performance, an inference pipeline has been developed. The model takes as input $256 \times 256 \times 3$ GeoTIFF images and outputs a classification report in the form of a pie graph and a post–processed image. In Figure 5.11 is shown the pipeline of the algorithm.

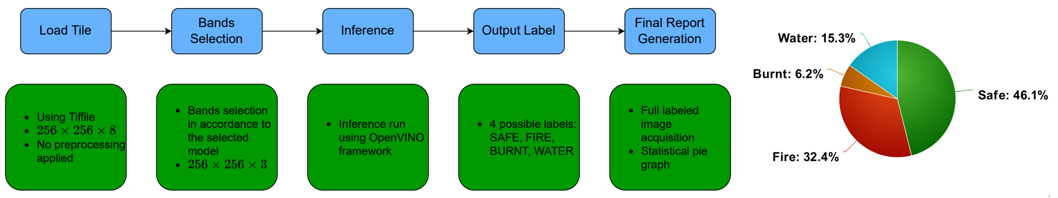


Figure 5.11. PhiFireAI Inference Pipeline.

Below the main processing steps are detailed:

- the first step is to load the input tile. Since the tile has a tiff extension, the Python library *tiffile* is used to read the image correctly. The image has dimension $256 \times 256 \times 8$;
- the next step is to select the correct sequence of bands in order to be consistent with the trained model. In this case three bands are selected: B1, B2 and B3;
- the processed image, which is now of dimension $256 \times 256 \times 3$, is passed as input to the model in order to perform the inference. The model returns as

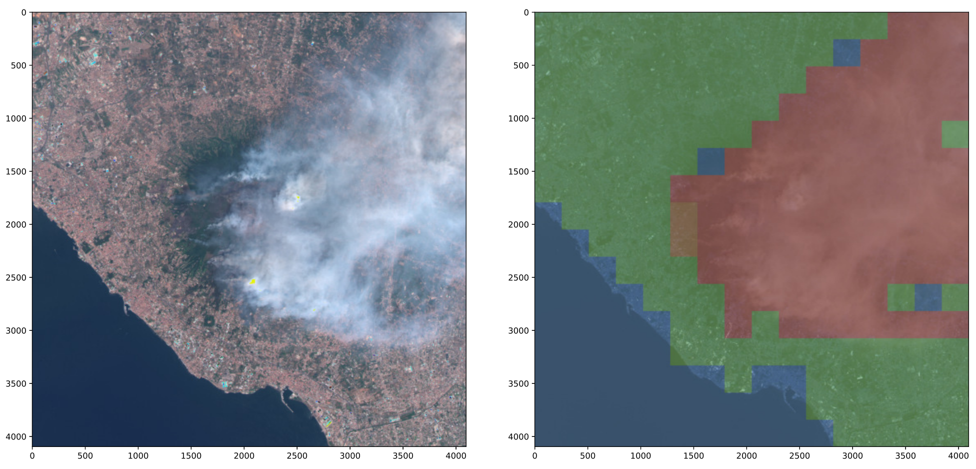
output the predicted label. There are four possible output values: 0 for Safe, 1 for Fire, 2 for Burnt and 3 for Water;

- after the inference is performed, all the predicted labels are used to obtain a full labeled acquisition image as the one in Fig. 5.12b and a pie graph. The latter shows the percentage of each class in the image. These two elements form the report which will be downlinked to the intervention centers.

This pipeline is used both for the test set validation and for the test over the space–qualified hardware.

5.4.3 Test Set Validation

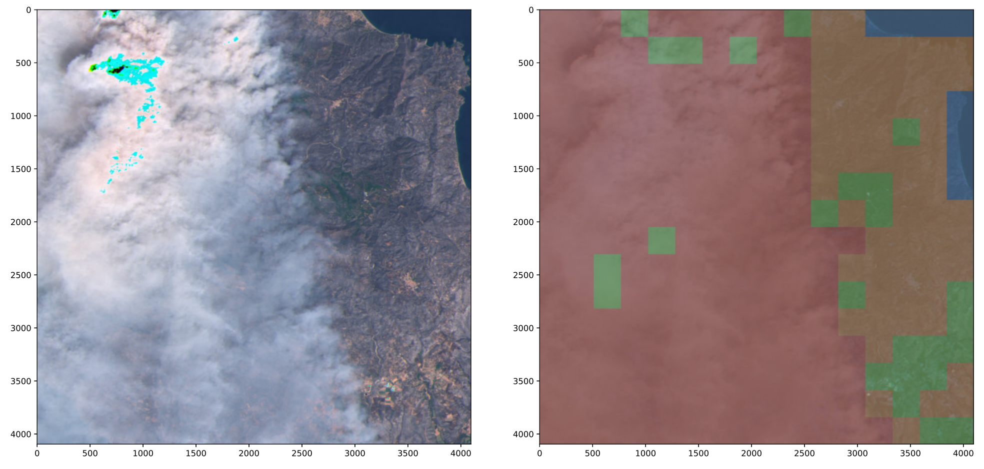
The test set is composed of 512 tiles, which correspond to two full acquisitions. The test set has been chosen in such a way to test the capability of the trained model to correctly detect and classify wildfire areas and, more in general, all the other class features. To do this, two images which include all the possible detectable areas (safe, fire, burnt and water areas) have been considered. In the test set definition phase, particular attention has been put in looking for images with the presence of large active wildfires.



(a) *Original Test Image.* **(b)** *Predicted Test Image.*

Figure 5.12. Inference Result for the first test image. In green is shown the safe area; in red the wildfire area is represented; in orange the burnt area is highlighted; in blue is shown the water area.

In Figures 5.12 and 5.13 are shown the results on the two test images. From these plots it can be seen how the model is able to correctly identify the wildfire area in both cases. In the second test image a large burnt area is also detected correctly. Water areas have also been detected correctly. Note also the presence of a some



(a) *Original Test Image.* **(b)** *Predicted Test Image.*

Figure 5.13. Inference result for the second test image. In green is shown the safe area; in red the wildfire area is represented; in orange the burnt area is highlighted; in blue is shown the water area.

outliers, i.e. wrongly classified tiles, in both cases. This is expected since, from the confusion matrix analysis, the percentage of FP and FN was not zero. However, the overall analysis of the full predicted image is able to give a clear overview of the situation in those areas. This would not be possible with an analysis performed tile by tile.

Along these data, a classification report is provided in order to give more information on the area occupied by the wildfire. It shows the percentage of each class with respect to the full acquisition image dimension. This report can be used to have a more clear idea of the extension of the wildfire areas.

5.4.4 Space–Qualified Test

The aim of this solution was to develop an algorithm to perform classification of wildfires areas on–board a satellite. This implies the model has to satisfy the constraints imposed by the satellite processing characteristics. In order to have a preliminary idea of the on–board performance of the solution, the inference pipeline has been executed on a space–qualified board, namely the Unibap's ix5–100 board.

This device's specifications have already been described in Section 3.3.2. This on–board processor is equipped, along with a CPU, a GPU and an FPGA, with a VPU which is exclusively dedicated to execute AI applications. The latter is based on Intel Movidius Myriad X AI accelerator.

To be compatible with the Myriad device, the model must be converted into a specific format, namely an xml file showing all the layers characteristics and a bin file which includes the model weights. This conversion has to be performed by using the Intel OpenVINO framework. In particular, the challenge's rules imposed the use of OpenVINO 2020.3 version, which was the one compatible with the Φsat–2 on–board processor.

The main steps for the conversion are detailed below:

1. model conversion from TensorFlow to ONNX format;
2. model optimization by scaling the model weights to FP16;
3. model conversion to IR format;
4. final xml and bin files generation.

The performance parameters considered in this test are:

- *latency*, which represents the time it takes to perform an inference request;
- *throughput*, which represents the number of frames processed per second. In this case each frame is represented by a single tile.

In the executed test, the processing of the images has been performed in a sequential way, which means that only one tile at a time is handled by the algorithm.

In Table 5.11 are shown the results of the test on the iX5–100 board. The test has been performed on the three candidate models. The inference time in the table is computed as the average time over all the processed tiles. Looking at the throughput it can be said that the time it takes to process a full acquisition is around 10 seconds. In the calculation of this parameter, the labelling of the tiles has also been taken into account along with other operations. In the real case, these could be handled by sending the data on Earth (i.e. all the processed tiles along with their predicted labels) in the case of wildfires detection (e.g. when multiple Fire–1 labels are predicted), performing the report generation offline. This will allow to further increase the throughput and, as consequence, the performances of the on–board algorithm.

model	**input size**	**avg inference time (seconds)**	**throughput (FPS)**
model–1	$256 \times 256 \times 3$	0.0260	24.4
model–2	$256 \times 256 \times 4$	0.0268	23.6
model–3	$256 \times 256 \times 3$	0.0260	24.5

Table 5.11. Average inference time and throughput of the three models.

5.5 Conclusions and Future Works

The proposed algorithm is able to detect correctly the presence of wildfires in a particular region and can be very useful to alert the authorities allowing them to intervene in time. It has also been tested over a space–qualified hardware, in order to analyze its performances for satellite on–board applications.

This idea can be further enhanced in terms of accuracy and generalization by increasing the number of data over which to train the model. An higher number of data allows to perform a more detailed analysis of the acquisition: indeed, a larger dataset may allow the model to be enriched by adding new classes, taking into account several environmental characteristics. Some useful features may be detecting the presence of towns or agricultural areas near the wildfires in order to notify the citizens of the danger and evacuate them in time. Another useful addition could be the classification of forests or other green areas. This allows the authorities to perform preventive measures to avoid possible danger to the environment.

The obtained results have been obtained by performing inferences in a sequential pattern. The OpenVINO framework offers the possibility of parallelizing the inference requests, feature which could be exploited in further studies to improve the on–board processing speed.

Chapter 6

Conclusions

In this work several space–based applications have been analyzed. The main focus was on on–board satellite navigation algorithms, with two different approaches which have been proposed.

The first one related to Earth navigation in which the ODTS estimation algorithm is aided by multiple GNSS measurements which offer great precision in position and velocity determination. A detailed analysis of satellite visibility was also performed in order to develop the EKF. The visibility algorithm has also been tested with real telemetry data and its outputs resulted very similar to the real data. The results have shown how the use of multiple GNSS measurements allows to obtain better performances also in the case of multiple error contributions in the measurement equations.

In the second approach, a first step over a Moon–based satellite on–board TRN system has been performed. Different from the previous case, non–standard measurements are needed due to the lack of data from Earth. AI comes in as a very good solution to this problem in the form of crater detection and matching algorithms. A benchmark of several NN architectures has been performed over a space–qualified hardware to check which solution would be able to satisfy on–board real–time constraints. The tests have been performed over a space–qualified processor. The results have shown that object detection architectures perform much better than image segmentation ones both in terms of latency and throughput.

Two additional satellite–based solutions have been studied and presented related to Free–space Satellite Optical communication and Earth Observation respectively. In the first one a mixed AI and RL approach has been proposed in order to generate an optimal data–path control law for point–to–point data exchange between two areas of Earth very far from each other. Its performances resulted to be the best among other standard benchmark solutions.

The second solution has been developed as an answer to an ESA challenge. In particular an AI algorithm for on–board wildfire detection in real–time by using CNNs has been proposed. The data were obtained by using the Φsat–2 satellite image simulator made available by ESA, starting from analyzing spatial and temporal wildfires data from a public database. The proposed solution ended up being selected as one of the top 5 applications and is now under a fine tuning process with the help of ESA experts.

Future works will be focused on mitigating and hopefully solving the limitations of all the proposed approaches.

Appendix

A Perturbation Models

A.1 Gravity Model

The total acceleration acting on the satellite due to the Earth's gravity is given by:

$$\mathbf{a}_{\text{GRAV}}(\mathbf{X}) = -\mu\frac{\mathbf{X}}{r^3} + \mathbf{a}_{\text{GRAV-NS}}(\mathbf{X}). \tag{A.1}$$

This is simply the acceleration due to the spherically symmetric mass of the Earth plus the non–spherical perturbation $\mathbf{a}_{\text{GRAV-NS}}$. μ is Earth's gravitational parameter. $\mathbf{X} = \begin{pmatrix} X_1 & X_2 & X_3 \end{pmatrix}^T$ is the position vector given in ECEF reference frame and $r = \|\mathbf{X}\|$ is the position magnitude.

The total gravitational acceleration vector is calculated as the first partial derivative of the potential function U w.r.t. $\mathbf{X}$:

$$\mathbf{a}_{\text{GRAV}}(\mathbf{X}) = \frac{\partial U}{\partial \mathbf{X}}. \tag{A.2}$$

The Earth gravity potential U is written as a series of harmonics in order to take into account the actual mass distribution of the Earth:

$$U = \frac{\mu}{r} + \sum_{n=2}^{\infty}\sum_{m=0}^{n}\frac{\mu}{r}\left(\frac{R}{r}\right)^n P_{n,m}(\epsilon)[C_{n,m}cos(m\lambda) + S_{n,m}sin(m\lambda)] \tag{A.3}$$

where

- n is the degree of each term: $2 < n < \infty$;
- m is the order of each term: $0 < m < n$;
- R is the Earth radius;
- $P_{n,m}$ are the associated Legendre functions;
- $\epsilon = X_3/r$ is the sine of latitude;

- $C_{n,m}$ and $S_{n,m}$ are the unnormalized cosine and sine gravity coefficients that result from the mass distribution of the planet (hey depend on the chosen model);

- λ is the longitude.

The sine of latitude ϵ and the longitude λ are given by:

$$\epsilon = \frac{X_3}{r} \tag{A.4}$$

$$\tan\lambda = \frac{X_2}{X_3}. \tag{A.5}$$

The term $\frac{\mu}{r}$ represents the spherical part of the potential: this would be Earth's gravitational potential if it was a perfect sphere. The $C_{2,0}$ coefficient corresponds to the J_2 perturbation.

The $C_{n,m}$and $S_{n,m}$ coefficients depend on the chosen model and are usually published in normalized form ($\bar{C}_{n,m}$ and $\bar{S}_{n,m}$). The unnormalized coefficients are used because the derivation of the gravity acceleration is somewhat simpler. The relationship between the normalized and the unnormalized form is given by:

$$C_{n,m} = N_{n,m}\bar{C}_{n,m} \tag{A.6}$$

$$S_{n,m} = N_{n,m}\bar{S}_{n,m} \tag{A.7}$$

where

$$N_{n,m} = \sqrt{\frac{(n-m)!(2n+1)(2-\sigma_{0m})}{(n+m)!}}, \quad \delta = \begin{cases} 1 & m = 0 \\ 0 & m \neq 0 \end{cases}. \tag{A.8}$$

It is also important to point out that the coefficients $C_{n,m}$ and $S_{n,m}$ are given in ECEF reference frame, so they need to be converted in ECI reference frame before being used in the equation of motion for orbital propagation.

In the following, some definitions and simplifications will be given in order to rewrite the Earth potential U in a more useful form. Defining ρ as:

$$\rho = X_1^2 + X_2^2 \tag{A.9}$$

the trigonometric functions c_m and s_m are defined in the following way:

$$c_m = \left(\frac{\rho}{r}\right)^m cos(m\lambda) \quad s_m = \left(\frac{\rho}{r}\right)^m sin(m\lambda). \tag{A.10}$$

The associated Legendre functions are given by:

$$P_{n,m}(\epsilon) = (1-\epsilon^2)^{\frac{m}{2}}\left(\frac{\partial^m P_n}{\partial \epsilon^m}\right) = \left(\frac{r^2 - X_3^2}{r^2}\right)^{\frac{m}{2}}\left(\frac{\partial^m P_n}{\partial \epsilon^m}\right) = \frac{\rho^m}{r^m} P_n^m \tag{A.11}$$

where P_m are the Legendre polynomials, $\epsilon = X_3/r$ and P_n^m are knokn as the unnormalized derived Legendre functions. The Earth gravitational potential can now be rewritten as

$$U = \frac{\mu}{r} + \sum_{n=2}^{\infty}\sum_{m=0}^{n} \frac{\mu}{r}\left(\frac{R}{r}\right)^n P_n^m (C_{n,m} c_m + S_{n,m} s_m). \tag{A.12}$$

This form is especially useful since P_n^m, c_m and s_m can be calculated recursively.

EGM2008 Model

The above spherical harmonic representation of the Geopotential model can use one of several different sets of coefficients $S_{n,m}$ and $C_{n,m}$. These sets of coefficients are matched to a given value of μ and R.

This model is complete to spherical harmonic degree and order 2159, contains additional coefficients extending to degree 2190 and order 2159 and is referenced to the WGS84 reference Ellipsoid. The data for the EGM2008 model were obtained with altimetry, with terrestrial data and with satellite data. The values of R and μ using the WGS84 ellipsoid are:

$$R = 6378136.3 \text{ [m]} \quad \mu = 3.986004418e14\,[\text{m}^3/\text{s}^2]. \tag{A.13}$$

The EGM2008 model can be used up to order and degree 120. This choice has been made because:

- In LEO, 120 is high enough to have precise estimation of the gravity acceleration. The accuracy on gravity acceleration obtained with degree and order 120 is significantly higher than the absolute value and accuracy of other perturbation models (especially the drag model because of the uncertainties on the atmosphere density);
- In higher orbits, like GEO, the gravity acceleration is smaller w.r.t. lower orbits because of the dependence on $1/r^2$. As a consequence, the absolute error made using less harmonics is lower on higher orbits;
- The use of unnormalized coefficients is limited to low degrees and orders: because of the $(n+m)!$ term at the denominator of (A.8), at a certain point (around degree 150) the unnormalized coefficients become equal to zero for the

current computer precisions. A lower number simply cannot be represented by the computer.

A.2 Magnetic Field Model

The Earth's magnetic field crudely resembles that of a central dipole. On the Earth's surface the field varies from being horizontal and of magnitude about 30000 nT near the equator to vertical and about 60000 nT near the poles; the root mean square (rms) magnitude of the vector over the surface is about 45 000 nT. The geomagnetic field also varies in time, on a time–scale of months and longer, in an as yet unpredictable manner. This so–called secular variation (SV) has a complicated spatial pattern, with a global rms magnitude of about 80 nT/year. Consequently, any numerical model of the geomagnetic field has to have coefficients which vary with time.

The total magnetic field vector in ECEF reference frame is calculated as the first partial derivative of the potential function V w.r.t. the ECEF position $\mathbf{X} = \begin{pmatrix} X_1 & X_2 & X_3 \end{pmatrix}^T$:

$$\mathbf{B} = -\frac{\partial V}{\partial \mathbf{X}}. \tag{A.14}$$

The potential V for the Earth magnetic field is written as a series of harmonics:

$$V = R \sum_{n=1}^{N} \sum_{m=0}^{n} \left(\frac{R}{r}\right)^{n+1} P^m_{n,\mathrm{SCHMIDT}}(cos\theta)[g^m_n cos(m\lambda) + h^m_n sin(m\lambda)] \tag{A.15}$$

where

- n is the degree of each term: $1 < n < N$. The maximum degree N depends on the chosen model;
- m is the order of each term: $0 < m < n$;
- R is the Earth radius. It depends on the chosen model;
- $P^m_{n,\mathrm{SCHMIDT}}$ are the Schmidt normalized associated Legendre functions;
- θ is the geodetic colatitude;
- g^m_n and h^m_n are the spherical harmonics coefficients. They depend on the chosen model;
- λ is the longitude.

The cosine of the colatitude $\cos\theta$ is equal to the sine of the latitude ϵ

$$cos\theta = \frac{X_3}{r} = \epsilon. \tag{A.16}$$

The longitude λ is given by:

$$\tan\lambda = \frac{X_3}{X_2}. \tag{A.17}$$

The Schmidt normalized associated Legendre functions $P^m_{n,\mathrm{SCHMIDT}}$ are related to the associated Legendre functions $P_{n,m}$ by:

$$P^m_{n,\mathrm{SCHMIDT}}(\epsilon) = N_{n,m}P_{n,m} \tag{A.18}$$

where:

$$N_{n,m} = \sqrt{\frac{2(n-m)!}{(n+m)!} - \delta_{0m}} \quad \delta = \begin{cases} 1 & m = 0 \\ 0 & m \neq 0 \end{cases} \tag{A.19}$$

is the Schmidt normalization factor.

Defining:

$$C_{n,m} = N_{n,m}g^m_n \tag{A.20}$$

$$S_{n,m} = N_{n,m}h^m_n \tag{A.21}$$

the magnetic field potential V becomes:

$$V = \sum_{n=1}^{N}\sum_{m=0}^{n} \frac{R^2}{r}\left(\frac{R}{r}\right)^n P^m_n(\epsilon)[C_{n,m}cos(m\lambda) + S_{n,m}sin(m\lambda)]. \tag{A.22}$$

In the following, some definitions and simplifications will be given in order to rewrite the Earth potential U in a more useful form [158]. Defining ρ as:

$$\rho = X_1^2 + X_2^2. \tag{A.23}$$

The trigonometric functions c_m and s_m are defined:

$$c_m = \left(\frac{\rho}{r}\right)^m cos(m\lambda) \tag{A.24}$$

$$s_m = \left(\frac{\rho}{r}\right)^m sin(m\lambda). \tag{A.25}$$

The associated Legendre functions are given by:

$$P_{n,m}(\epsilon) = (1-\epsilon^2)^{\frac{m}{2}}\left(\frac{\partial^m P_n}{\partial\epsilon^m}\right) \tag{A.26}$$

where P_m are the Legendre polynomials. Using (A.16) for the definition of ϵ, the previous equation can be rewritten as:

$$P_{n,m}(\epsilon) = \Big(\frac{r^2 - X_3^2}{r^2}\Big)^{\dfrac{m}{2}} \Big(\frac{\partial^m P_n}{\partial \epsilon^m}\Big) = \frac{\rho^m}{r^m} P_n^m \tag{A.27}$$

where P_n^m are known as the unnormalized derived Legendre functions:

$$P_n^m = \frac{\partial^m P_n}{\partial \epsilon^m}. \tag{A.28}$$

Separating the $n = 1$ term from (A.25) and (A.27), the magnetic field potential V can be rewritten as:

$$V = \frac{R^3}{r^2}\Big[P_1^0 C_{1,0} + \Big(C_{1,1}\frac{X_1}{r} + S_{1,1}\frac{X_2}{r}\Big)P_1^1\Big] + \sum_{n=2}^{N}\sum_{m=0}^{n}\frac{R^2}{r}\Big(\frac{R}{r}\Big)^n P_n^m (C_{n,m}c_m + S_{n,m}s_m). \tag{A.29}$$

This form is especially useful since P_n^m, c_m and s_m can be calculated recursively.

IGRF Model

The International Geomagnetic Reference Field (IGRF) model is released by the International Association of Geomagnetism and Aeronomy (IAGA). The model is updated at 5 years intervals. The IGRF is an attempt by IAGA to provide an easily–usable model acceptable to a variety of users. It is meant to give a reasonable approximation, near and above the Earth's surface, to that part of the Earth's magnetic field which has its origin inside the surface. At any one epoch, the IGRF specifies the numerical coefficients of a truncated spherical harmonic series: for dates until 2000 the truncation is at N=10, with 120 coefficients, but from 2000 the truncation is at N=13, with 195 coefficients. Such a model is specified every 5 years, for epochs 1900.0, 1905.0 etc. For dates between the model epochs, coefficient values are given by linear interpolation.

WMM Model

The World Magnetic Model (WMM) is the standard model used by the U.S. Department of Defense, the U.K. Ministry of Defence, the North Atlantic Treaty Organization (NATO) and the International Hydrographic Organization (IHO), for navigation, attitude and heading referencing systems using the geomagnetic field. It is also used widely in civilian navigation and heading systems. The model is produced at 5–year intervals, with the current model (as of 2020) expiring on December 31, 2024. The WMM consists of a degree and order 12 spherical–harmonic main (i.e., core–generated) field model comprised of 168 spherical–harmonic Gauss coefficients and degree and order 12 spherical–harmonic Secular–Variation (SV) (core–generated, slow temporal variation) field model. Provided that suitable satellite magnetic observations are available, the prediction of the WMM is highly accurate on its release date and then subsequently deteriorates towards the end of the 5–year epoch, when it has to be updated with revised values of the model coefficients.

The value for the reference Earth radius R for both models is $R = 6372.2$ km.

The WMM and the IGRF are estimated from the most recent data and are of comparable quality. The differences between IGRF and WMM are within expected model inaccuracy. The WMM is a predictive–only model and is valid for the current five–year time span. The IGRF, on the other hand, is also retrospectively updated.

A.3 Third Boby Model

According to Newton's law of gravity, the acceleration of a satellite by a point mass M is given by:

$$\mathbf{a}_{\mathrm{M}\rightarrow\mathrm{SC}} = GM\frac{\mathbf{d}-\mathbf{r}}{\|\mathbf{d}-\mathbf{r}\|^3} \tag{A.30}$$

where $\mathbf{r}$ and $\mathbf{d}$ are the respectively geocentric coordinates of the satellite and of M, and G is the gravitational constant.

Some care is required, however, before this expression can be used for describing the satellite's motion with respect to the center of the Earth. The value of $\mathbf{a}_{\mathrm{M}\rightarrow\mathrm{SC}}$ in the equation refers to an inertial or Newtonian coordinate system in which the Earth is not at rest, but is itself subject to the following acceleration due to M:

$$\mathbf{a}_{\mathrm{M}\rightarrow\mathrm{EARTH}} = \frac{\mathbf{d}}{\|\mathbf{d}\|^3}. \tag{A.31}$$

Both these values have to be subtracted to obtain the third body acceleration [12]:

$$\mathbf{a}_{\mathrm{TB}} = \ddot{\mathbf{r}} = GM\left(\frac{\mathbf{d}-\mathbf{r}}{\|\mathbf{d}-\mathbf{r}\|^3} - \frac{\mathbf{d}}{\|\mathbf{d}\|^3}\right). \tag{A.32}$$

A.4 Atmospheric Drag Model

The acceleration of the satellite a_{DRAG} due to atmospheric drag expressed in ECI reference frame is modeled using the following equation:

$$a_{\mathrm{DRAG}} = -\frac{1}{2}\rho Cd\frac{A}{m}V_r\mathbf{V}_r \tag{A.33}$$

where

- Cd is the drag coefficient which is expected to have a value in the range of about 2 to 3 with 2 being a typical reference value for a spherical spacecraft;

- ρ is the atmospheric density whose expression depends on the chosen model;
- A is the SC equivalent drag surface;
- m is the SC mass;
- $\mathbf{V}_r$ is the velocity of the spacecraft relative to Earth's atmosphere and V_r is its norm.

$\mathbf{V}_r$ is computed in the ECI frame under the assumption that the atmosphere rotates with the Earth. It is given by the difference between the spacecraft velocity VEL_ECI and the atmosphere velocity $W\_ECI \times POS\_ECI$:

$$\mathbf{V}_r = VEL\_ECI - W\_ECI \times POS\_ECI \tag{A.34}$$

where POS_ECI is the SC position in ECI reference frame and W_ECI is the Earth angular velocity vector expressed in ECI reference frame.

For the density ρ several models are available. It usually is function of the ECEF position of the spacecraft, the date and the Sun position. The most used in literature is the Modified Harris–Priester model.

Modified Harris–Priester Model

The atmospheric density ρ is calculated using an analytic approximation to the Harris–Priester atmospheric model, as described in [12]. The Harris–Priester model is based on the properties of the upper atmosphere as determined from the solution of the heat conduction equation under quasi–hydrostatic conditions. A modified version of the Harris–Priester model is used: this modification attempts to account for the diurnal bulge by including a cosine variation between a maximum density profile at the apex of the diurnal bulge and a minimum profile at the antapex of the diurnal bulge. As the atmospheric heating due to the solar radiation leads to a gradual increase of the atmospheric density, the apex of this bulge is delayed by approximately 2 hours, equivalent to a location 30 deg to the east of the subsolar point.

In this model, the density is function of the altitude h above the reference ellipsoid WGS84 and the Sun coordinates which are needed to calculate the diurnal bulge direction uB_ECEF. The altitude h depends on the ECEF position POS_ECEF of the spacecraft and is given in [159]:

$$h = r - \frac{a(1-f)}{\sqrt{1 - f(2-f)(1-s^2)}} \tag{A.35}$$

where:

- r is the norm of the vector POS_ECEF of the SC;
- a is the reference radius of the WGS84 Ellipsoid;
- f is the flattening of the WGS84 Ellipsoid;
- s is the z–coordinate of the normalized POS_ECEF, so that $\sqrt{1-s^2}$ is the cosine of the declination of the satellite.

The antapex and apex density $\rho_{min}(h)$ and $\rho_{max}(h)$ at the altitude h are computed through the exponential interpolation between tabulated minimum and maximum density values $\rho_{min}(h_i)$ and $\rho_{max}(h_i)$. The Harris–Priester model gives a table of 50 altitudes h_i between 100 km and 1000 km with the values of the maximum and minimum densities at those altitudes. For the altitude $h_i \leq h \leq h_{i+1}$ the exponential interpolation is given by:

$$\rho_{min}(h) = \rho_{min}(h_i) * exp\Big(\frac{h_i - h}{H_m}\Big) \tag{A.36}$$

$$\rho_{max}(h) = \rho_{max}(h_i) * exp\Big(\frac{h_i - h}{H_M}\Big) \tag{A.37}$$

with:

$$H_m(h) = \frac{h_i - h_{i+1}}{ln[\rho_{min}(h_{i+1})/\rho_{min}(h_i)]} \tag{A.38}$$

$$H_M(h) = \frac{h_i - h_{i+1}}{ln[\rho_{max}(h_{i+1})/\rho_{max}(h_i)]}. \tag{A.39}$$

The direction of the diurnal bulge uB_ECEF is obtained from the direction of the Sun in ECEF frame $SUN\_DIR = SUN\_ECEF/norm(SUN\_ECEF)$ as follows:

$$uB\_ECEF(1) = cos(\delta)cos(\alpha+\lambda) = SUN\_DIR(1)cos(\lambda) - SUN\_DIR(2)sin(\lambda) \tag{A.40}$$

$$uB\_ECEF(2) = cos(\delta)sin(\alpha+\lambda) = SUN\_DIR(1)sin(\lambda) - SUN\_DIR(2)cos(\lambda) \tag{A.41}$$

$$uB\_ECEF(3) = sin(\delta) = SUN\_DIR(3) \tag{A.42}$$

where α and δ are respectively the right ascension and the declination of the Sun in ECEF reference frame and λ is the lag angle in longitude which is equal to 30 deg.

Finally the density ρ is given by:

$$\rho = \rho_{min} + (\rho_{max} - \rho_{min})\Big[\frac{1}{2} + \frac{1}{2}uB\_ECEF\Big(\frac{POS\_ECEF}{r}\Big)\Big]^{\frac{n}{2}} \tag{A.43}$$

where n is the drag precision parameter which has a numerical value of 2 for low inclination orbits and 6 for polar orbits.

A.5 Solar Radiation Pressure Model

A satellite that is exposed to solar radiation experiences a small force that arises from the absorption or reflection of photons. The acceleration due to the solar radiation depends on the satellite's mass and surface area. The following description of the solar radiation pressure is based on [12].

The size of the solar radiation pressure is determined by the solar flux:

$$\Phi = \frac{\Delta E}{A\Delta t} \tag{A.44}$$

where ΔE is the energy that passes through an area A in the time interval Δt. A single photon of energy E_ν carries an impulse p_ν:

$$p_\nu = \frac{E_\nu}{c} \tag{A.45}$$

where c is the speed of light. Accordingly, the total impulse of an absorbing body that is illuminated by the Sun changes by Δp during the time Δt:

$$\Delta p = \frac{\Delta E}{c} = \frac{\Phi}{c}A\Delta t. \tag{A.46}$$

Assuming that the satellite's surface A absorbs all photons and is perpendicular to the incoming radiation, the force experienced by the satellite is given by:

$$F = \frac{\Delta p}{\Delta t} = \frac{\Phi}{c}A = PA. \tag{A.47}$$

At a distance of one Astronomical Unit from the Sun the solar flux is:

$$\Phi \approx 1367\,[\mathrm{Wm}^{-2}]. \tag{A.48}$$

The reference solar radiation pressure P_{Ref} at one Astronomical Unit from the Sun is given by:

$$P_{\mathrm{Ref}} = \frac{\Phi}{A} \approx 4.56 \cdot 10^{-6}\,[\mathrm{Nm}^{-2}] \tag{A.49}$$

The general case of a satellite surface with an arbitrary orientation is illustrated

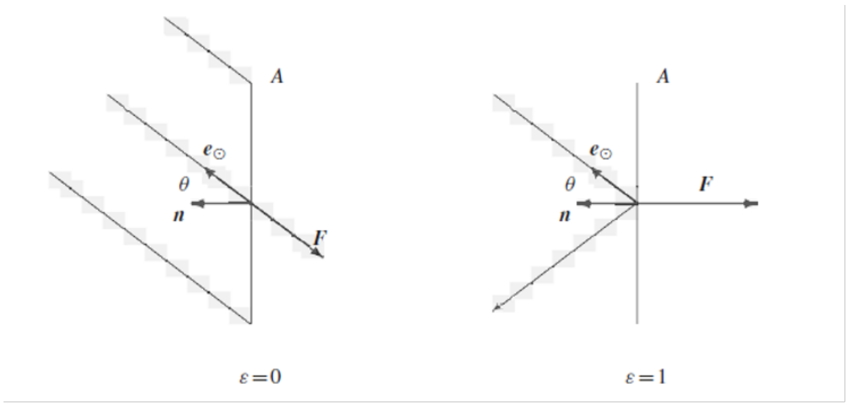


Figure A.1. Force due to solar radiation pressure for absorbing ($\epsilon = 0$) and reflecting ($\epsilon = 1$) surface elements.

in Figure A.1.

In contrast to specular reflection, the diffuse reflection of light is neglected in the sequel. The normal unit vector **n** gives the orientation of the surface A. It is inclined at an angle θ to the unit vector **e** which points into the direction of the Sun.

For an absorbing surface, the force $\mathbf{F}_{\text{abs}}$ is directed away from the Sun and is equal to:

$$\mathbf{F}_{\text{abs}} = -P_{\text{Ref}} \cos\theta A \mathbf{e} \tag{A.50}$$

where $\cos\theta A$ is the cross–section of the bundle of light that illuminates A.

For a reflecting surface, the force is not, in general, directed away from the Sun, since no impulse is transferred in the direction parallel to the surface. Due to the reflected light rays, the impulse transferred in the direction of **n** is twice as large w.r.t. the case of pure absorption:

$$\mathbf{F}_{\text{refl}} = -2P_{\text{Ref}} \cos\theta A \cos\theta \mathbf{n}. \tag{A.51}$$

These two equations can be combined for a body that reflects a fraction ϵ of the incoming radiation ΔE, while it absorbs the remaining energy $(1-\epsilon)\Delta E$:

$$\mathbf{F} = -P_{\text{Ref}} \cos\theta A[(1-\epsilon)\mathbf{e} + 2\epsilon \cos\theta \mathbf{n}]. \tag{A.52}$$

For typical materials used in the construction of satellites, the reflectivity ϵ lies in the range from 0.2 to 0.9. Due to the eccentricity of the Earth's orbit, the distance between an Earth–orbiting satellite and the Sun varies between $147 \cdot 10^6$ km and $152 \cdot 10^6$ km during the course of a year. This results in an annual variation of the reference solar radiation pressure by about $\pm 3.3\%$, since the solar flux decreases with the square of the distance from the Sun. Accounting for this dependence, the

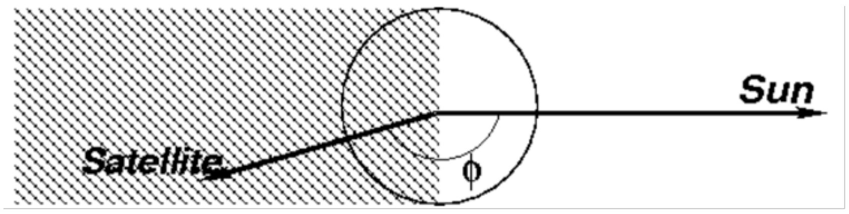


Figure A.2. Simple cilinder model for shadow of Earth.

following expression is finally obtained for the acceleration $\mathbf{a}_{\mathrm{SP}}$ of a satellite due to the solar radiation pressure:

$$\mathbf{a}_{\mathrm{SP}} = -P_{\mathrm{Ref}} \frac{d_{\mathrm{Ref}}^2}{r_{\mathrm{SUN}}^2} \frac{A}{m} \cos\theta[(1-\epsilon)\mathbf{e} + 2\epsilon\cos\theta\mathbf{n}] \tag{A.53}$$

where m is the mass of the satellite and r_{SUN} is the magnitude of the satellite to Sun vector $\mathbf{r}_{\mathrm{SUN}}$.

For many applications (e.g. satellites with large solar arrays) it suffices, however, to assume that the surface normal $\mathbf{n}$ points in the direction of the Sun $\mathbf{e}$. In this case, the previous equation may further be simplified, yielding:

$$\mathbf{a}_{\mathrm{SP}} = -P_{\mathrm{Ref}} C_{\mathrm{sp}} \frac{A}{m} \frac{\mathbf{r}_{\mathrm{SUN}}}{r_{\mathrm{SUN}}^3} d_{\mathrm{Ref}}^2 \tag{A.54}$$

where the radiation pressure coefficient C_{sp} stands for:

$$C_{\mathrm{sp}} = 1 + \epsilon. \tag{A.55}$$

If the satellite is in eclipse condition, no solar radiation pressure force acts on the satellite. In order to evaluate the eclipse condition, a simple cylinder model has been implemented (see Figure A.2).

Given the Sun position unity vector $\mathbf{e}$, the satellite's position vector $\mathbf{r}$ and the Earth's radius R, and defining:

$$F_1 = r\cos\Phi = \mathbf{r}^T\mathbf{e} \tag{A.56}$$

$$F_2 = r\sqrt{1-\cos^2\Phi} = \|\mathbf{r} - F_1\mathbf{e}\|. \tag{A.57}$$

The eclipse condition is true if:

$$F_1 < 0 \quad \& \quad F_2 < R \tag{A.58}$$

B Algorithms

B.1 GDOP Computation

As a first step in computing DOP, consider the unit vectors from the receiver to the i–th satellite:

$$\left(\frac{x_i - x}{R_i} \quad \frac{y_i - y}{R_i} \quad \frac{z_i - z}{R_i} \right) \tag{B.59}$$

where x_i, y_i and z_i are the coordinates of the i–th satellite, x, y and z are the coordinates of the receiver, R_i is computed as:

$$R_i = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2}. \tag{B.60}$$

After this the matrix $\mathbf{A}$ must be formulated. Considering N pseudorange measurements the matrix assumes this form:

$$\mathbf{A} = \begin{bmatrix} \frac{x_1 - x}{R_1} & \frac{y_1 - y}{R_1} & \frac{z_1 - z}{R_1} & 1 \\ \frac{x_2 - x}{R_2} & \frac{y_2 - y}{R_2} & \frac{z_2 - z}{R_2} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ \frac{x_N - x}{R_N} & \frac{y_N - y}{R_N} & \frac{z_N - z}{R_N} & 1 \end{bmatrix}. \tag{B.61}$$

The first three elements of each row of $\mathbf{A}$ are the components of a unit vector from the receiver to the indicated satellite. The last element of each row refers to the partial derivative of pseudorange w.r.t. receiver's clock bias. From this, one can formulate the covariance matrix $\mathbf{Q}$ as:

$$\mathbf{Q} = (\mathbf{A}^T \mathbf{A})^{-1} \tag{B.62}$$

which has the following form:

$$\mathbf{Q} = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{xz} & \sigma_{xt} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{yz} & \sigma_{yt} \\ \sigma_{xz} & \sigma_{yz} & \sigma_z^2 & \sigma_{zt} \\ \sigma_{xt} & \sigma_{yt} & \sigma_{zt} & \sigma_t^2 \end{bmatrix}. \tag{B.63}$$

The PDOP, TDOP and GDOP are give by:

$$\text{PDOP} = \sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_z^2} \tag{B.64}$$

$$\text{TDOP} = \sqrt{\sigma_t^2} \tag{B.65}$$

$$\text{GDOP} = \sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_z^2 + \sigma_t^2}. \tag{B.66}$$

So the GDOP is the square root of the trace of the $\mathbf{Q}$ matrix.

B.2 UDU Covariance Factorization

Given a generic covariance matrix $P \in \mathbb{R}^{N\times N}$ which is symmetric and positive definite, the algorithm returns two matrices $U \in \mathbb{R}^{N\times N}$ and $D \in \mathbb{R}^{N\times N}$ such that:

$$P = UDU^T \tag{B.67}$$

with:

$$U = \begin{bmatrix} 1 & U_{1,2} & . & . & . & . & . & . & U_{1,N} \\ 0 & 1 & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & 1 & U_{N-1,N} \\ 0 & . & . & . & . & . & . & 0 & 1 \end{bmatrix} \tag{B.68}$$

$$D = \begin{bmatrix} D_{1,1} & 0 & . & . & . & . & . & . & 0 \\ 0 & D_{2,2} & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & D_{N-1,N-1} & 0 \\ .0 & . & . & . & . & . & . & 0 & D_{N,N} \end{bmatrix}. \tag{B.69}$$

For the N–th coloumn:

$$D_{N,N} = P_{N,N} \tag{B.70}$$

$$U_{i,N} = \begin{cases} 1 & i = N \\ P_{i,N}/D_{N,N} & i = N-1, N-2, \dots, 1 \end{cases}. \tag{B.71}$$

Then, for the remaining coloumns $j = N-1, \dots, 1$, compute:

$$D_{j,j} = P_{j,j} - \sum_{k=j+1}^{N} [D_{k,k} U_{j,k}^2] \tag{B.72}$$

$$U_{i,j} = \begin{cases} 0 & i > j \\ 1 & i = j \\ \left[P_{i,j} - \sum_{k=j+1}^{N} D_{k,k} U_{i,k} U_{j,k}\right] / D_{j,j} & i = j-1, j-2, \ldots, 1 \end{cases} . \tag{B.73}$$

B.3 UDU Covariance Propagation

When using UDU factorization of the state error covariance matrix, U and D are propagated directly. The covariance matrix can be reformed from the propagated U and D. The propagation requires the state transition matrix Φ and the state noise covariance matrix Q which can be factorized in G_d, Q_d such that:

$$Q = G_d Q_d G_d^T . \tag{B.74}$$

Note that in the filter algorithm the state noise covariance matrix is defined at the beginning and does not change during the iterations.

Given the matrices $U(t_i)$, $D(t_i)$, G_d, Q_d and the state transition matrix Φ define a $N \times 2N$ matrix Y in this way:

$$Y = \begin{bmatrix} \Phi U(t_i) & G_d \end{bmatrix} . \tag{B.75}$$

Define a $2N \times 2N$ diagonal matrix $\tilde{D}$ as:

$$\tilde{D} = \begin{bmatrix} D(t_i) & 0 \\ 0 & Q_d \end{bmatrix} . \tag{B.76}$$

In the following algorithm $\mathbf{a}_j$, $\mathbf{a}_k$, $\mathbf{c}_k$ and $\mathbf{d}_k$ each represent N vectors with 2N elements, not 2N $\times$ 2N matrices; and c_{kj} and a_{kj} represent element (j) in vector (k), not matrix element (k,j).

N vectors, each of dimension 2N, are initialized as follows:

$$\begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \ldots\ldots & \mathbf{a}_N \end{bmatrix} = Y^T \tag{B.77}$$

and iterated on the following relations for $k = N, N-1, \ldots, 1$:

$$\mathbf{c}_k = \tilde{D} \mathbf{a}_k . \tag{B.78}$$

Because $\tilde{D}$ is diagonal, $\mathbf{c}_k$ is computed as shown below:

$$c_{k,j} = \tilde{D}_{j,j} a_{k,j} \quad j = 1, 2, \ldots, 2N \tag{B.79}$$

$$D_{k,k}(t_{i+1}) = \mathbf{a}_k^T \mathbf{c}_k \tag{B.80}$$

$$\mathbf{d}_k = \frac{\mathbf{c}_k}{D_{k,k}(t_{i+1})} \tag{B.81}$$

$$\left.\begin{aligned} U_{j,k}(t_{i+1}) &= a_j^T d_k \\ a_j &\leftarrow a_j - U_{j,k}(t_i + 1) a_k \end{aligned}\right\} \quad j = 1, 2, \ldots, k-1 \tag{B.82}$$

where $\rightarrow$ denotes replacement or writing over. On the last iteration, for k=1, only the equations for $c_{k,j}$ and $D_{k,k}$ are computed.

B.4 UDU Measurement Update

Given $\hat{\mathbf{X}}^-$ the state before the correction, U^-, D^- the factorization matrices before the correction, $H \in \mathbb{R}^{1\times N}$ the jacobian matrix of the i–th measurement, Y the i–th measurement, $\hat{Y}$ the estimated i–th measurement and $y = Y - \hat{Y}$ the i–th measurement residual compute:

$$\mathbf{f} = [U^-]^T H^T \tag{B.83}$$

$$v_j = D_{i,j}^- f_j \quad j = 1, 2, \ldots, N \tag{B.84}$$

and set $a_0 = \sigma^2$, where σ is the 1 sigma measurement error.
For $j = 1, 2, \ldots, N$ compute:

$$a_j = a_{j-1} + f_j v_j. \tag{B.85}$$

The predicted measurement residual variance, V_k, is computed as:

$$V_k = a_N. \tag{B.86}$$

Then perform the n–sigma measurement residual test:

$$D = \frac{y}{\sqrt{V_k}}. \tag{B.87}$$

If $|D| \leq N_\sigma$, accept the measurement and continue the update, otherwise reject the measurement and restart with a new one. If the test is passed, update the covariance factors for $k = 1, 2, \ldots, N$ as follows:

$$D_{k,k}^+ = D_{k,k}^- a_{k-1}/a_k \tag{B.88}$$

$$b_k \leftarrow v_k \tag{B.89}$$

$$p_k = -f_k/a_{k-1} \tag{B.90}$$

$$\left.\begin{aligned} U^+_{j,k} &= U^-_{j,k} + b_j p_k \\ b_j &\leftarrow b_j + U^-_{j,k} v_k \end{aligned}\right\} \quad j = 1, 2, \ldots, k-1. \tag{B.91}$$

Then the Kalman gain vector $K \in \mathbb{R}^{N\times 1}$ is computes as:

$$K = \mathbf{b}/V_k. \tag{B.92}$$

The state is then updated:

$$\hat{\mathbf{X}}^+ = \hat{\mathbf{X}}^- + Ky \tag{B.93}$$

This procedure is performed for every measurement by replacing $U^- \leftarrow U^+$, $D^- \leftarrow D^+$ and $\hat{\mathbf{X}}^- \leftarrow \hat{\mathbf{X}}^+$.

Bibliography

[1] J. R. Vetter, “Fifty years of orbit determination,” *Johns Hopkins APL technical digest*, vol. 27, no. 3, p. 239, 2007.

[2] D. Menegatti, A. Giuseppi, and A. Pietrabissa, “Model predictive control for collision-free spacecraft formation with artificial potential functions,” in *2022 30th Mediterranean Conference on Control and Automation (MED)*, pp. 564–570, IEEE, 2022.

[3] A. Giuseppi, A. Pietrabissa, S. Cilione, and L. Galvagni, “Feedback linearization-based satellite attitude control with a life-support device without communications,” *Control Engineering Practice*, vol. 90, pp. 221–230, 2019.

[4] Y. Feng, “An alternative orbit integration algorithm for gps-based precise leo autonomous navigation,” *GPS Solutions*, vol. 5, pp. 1–11, 2001.

[5] R. C. Hart, K. R. Hartman, A. C. Long, T. Lee, and D. H. Oza, “Global positioning system (gps) enhanced orbit determination experiment (geode) on the small satellite technology initiative (ssti) lewis spacecraft,” *Proceedings of the ION-GPS-1996, Kansas City, MO, USA*, pp. 17–20, 1996.

[6] E.-J. Choi, J.-C. Yoon, B.-S. Lee, S.-Y. Park, and K.-H. Choi, “Onboard orbit determination using gps observations based on the unscented kalman filter,” *Advances in Space Research*, vol. 46, no. 11, pp. 1440–1450, 2010.

[7] R. Zanetti and C. D’Souza, “Recursive implementations of the schmidt-kalman ‘consider’filter,” *The Journal of the Astronautical Sciences*, vol. 60, no. 3-4, pp. 672–685, 2013.

[8] M. O. Karslioglu, E. Erdogan, and O. Pamuk, “Gps-based real-time orbit determination of low earth orbit satellites using robust unscented kalman filter,” *Journal of Aerospace Engineering*, vol. 30, no. 6, p. 04017063, 2017.

[9] S. Kavitha, P. Mula, M. Kamat, S. Nirmala, and J. G. Manathara, "Extended kalman filter-based precise orbit estimation of leo satellites using gps range measurements," *IFAC-PapersOnLine*, vol. 55, no. 1, pp. 235–240, 2022.

[10] A. Tantucci, A. Wrona, and A. Pietrabissa, "Precise orbit determination on leo satellite using pseudorange and pseudorange-rate measurements," in *2023 31st Mediterranean Conference on Control and Automation (MED)*, pp. 341–347, IEEE, 2023.

[11] N. K. Pavlis, S. A. Holmes, S. C. Kenyon, and J. K. Factor, "The development and evaluation of the earth gravitational model 2008 (egm2008)," *Journal of geophysical research: solid earth*, vol. 117, no. B4, 2012.

[12] O. M. E. Gill and O. Montenbruck, *Satellite orbits.* Springer, 2013.

[13] R. E. Kalman *et al.*, "Contributions to the theory of optimal control," *Bol. soc. mat. mexicana*, vol. 5, no. 2, pp. 102–119, 1960.

[14] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[15] R. E. Kalman and R. S. Bucy, "New results in linear filtering and prediction theory," 1961.

[16] G. J. Bierman and C. L. Thornton, "Numerical comparison of kalman filter algorithms: Orbit determination case study," *Automatica*, vol. 13, no. 1, pp. 23–35, 1977.

[17] G. J. Bierman, *Factorization methods for discrete sequential estimation.* Courier Corporation, 2006.

[18] C. L. Thornton, "Triangular covariance factorizations for," tech. rep., 1976.

[19] S. Evans, W. Taber, T. Drain, J. Smith, H.-C. Wu, M. Guevara, R. Sunseri, and J. Evans, "Monte: the next generation of mission design and navigation software," *CEAS Space Journal*, vol. 10, pp. 79–86, 2018.

[20] G. H. Golub and C. F. Van Loan, *Matrix computations.* JHU press, 2013.

[21] M. Verhaegen and P. Van Dooren, "Numerical aspects of different kalman filter implementations," *IEEE Transactions on Automatic Control*, vol. 31, no. 10, pp. 907–917, 1986.

[22] C. L. THORNTON and G. J. BIERMAN, "Gram-schmidt algorithms for covariance propagation," *International Journal of Control*, vol. 25, no. 2, pp. 243–260, 1977.

[23] O. Montenbruck and P. Ramos-Bosch, "Precision real-time navigation of leo satellites using global positioning system measurements," *GPS solutions*, vol. 12, pp. 187–198, 2008.

[24] T. Carrico, J. Carrico, L. Policastri, and M. Loucks, "Investigating orbital debris events using numerical methods with full force model orbit propagation," *Adv. Astronaut. Sci*, vol. 130, no. PART 1, pp. 407–426, 2008.

[25] CelesTrak, "Norad gp element sets current data." `https://celestrak.org/NORAD/elements/`. Accessed: 2023-09-09.

[26] W. J. Larson, J. R. Wertz, *et al.*, *Space mission analysis and design*, vol. 3. Springer, 1992.

[27] W. Marquis, "The gps block iir/iir-m antenna panel pattern rev 3," *LMCO 2013*, 2013.

[28] P. J. Teunissen and O. Montenbruck, *Springer handbook of global navigation satellite systems*, vol. 10. Springer, 2017.

[29] G. Seeber, *Satellite geodesy.* Walter de gruyter, 2003.

[30] C. Zucca and P. Tavella, "The clock model and its relationship with the allan and related variances," *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, vol. 52, no. 2, pp. 289–296, 2005.

[31] L. Galleani, L. Sacerdote, P. Tavella, and C. Zucca, "A mathematical model for the atomic clock error," *Metrologia*, vol. 40, no. 3, p. S257, 2003.

[32] E. D. Kaplan and C. Hegarty, *Understanding GPS/GNSS: principles and applications.* Artech house, 2017.

[33] L. Galleani, "A tutorial on the two-state model of the atomic clock noise," *Metrologia*, vol. 45, no. 6, p. S175, 2008.

[34] J. Kouba, "Improved relativistic transformations in gps," *GPS Solutions*, vol. 8, pp. 170–180, 2004.

[35] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[36] A. Zell, *Simulation neuronaler netze*, vol. 1. Addison-Wesley Bonn, 1994.

[37] S. Linnainmaa, *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors.* PhD thesis, Master's Thesis (in Finnish), Univ. Helsinki, 1970.

[38] D. E. Rumelhart, G. E. Hinton, R. J. Williams, *et al.*, “Learning internal representations by error propagation,” 1985.

[39] R. Venkatesan and B. Li, *Convolutional neural networks in visual computing: a concise guide.* CRC Press, 2017.

[40] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural networks*, vol. 61, pp. 85–117, 2015.

[41] N. Khan-Mayberry, “The lunar environment: Determining the health effects of exposure to moon dusts,” *Acta Astronautica*, vol. 63, no. 7-10, pp. 1006–1014, 2008.

[42] J. Capolicchio, T. Catuogno, C. Lauro, M. Eleuteri, and C. Stallo, “The bright side of the moon: Multi-constrained optimized maps in support of future mission planning,” in *72th International Astronautical Congress*, 2021.

[43] D. Smith, M. Zuber, G. Neumann, F. Lemoine, M. Robinson, O. Aharonson, J. Head, X. Sun, J. Cavanaugh, and G. Jackson, “The lunar orbiter laser altimeter (lola) on the lunar reconnaissance orbiter,” in *AGU Fall Meeting Abstracts*, vol. 2006, pp. U41C–0826, 2006.

[44] R. Povilaitis, M. Robinson, C. Van der Bogert, H. Hiesinger, H. Meyer, and L. Ostrach, “Crater density differences: Exploring regional resurfacing, secondary crater populations, and crater saturation equilibrium on the moon,” *Planetary and Space Science*, vol. 162, pp. 41–51, 2018.

[45] J. W. Head III, C. I. Fassett, S. J. Kadish, D. E. Smith, M. T. Zuber, G. A. Neumann, and E. Mazarico, “Global distribution of large lunar craters: Implications for resurfacing and impactor populations,” *science*, vol. 329, no. 5998, pp. 1504–1507, 2010.

[46] S. J. Robbins, “A new global database of lunar impact craters> 1–2 km: 1. crater locations and sizes, comparisons with published databases, and global analysis,” *Journal of Geophysical Research: Planets*, vol. 124, no. 4, pp. 871–892, 2019.

[47] N. Petro, J. Keller, B. Cohen, and T. McClanahan, “Ten years of the lunar reconnaissance orbiter (lro): Advancing lunar science and context for future

lunar exploration," in *50th Annual Lunar and Planetary Science Conference*, no. 2132, p. 2780, 2019.

[48] E. Mazarico, D. Rowlands, G. Neumann, D. Smith, M. Torrence, F. Lemoine, and M. Zuber, "Orbit determination of the lunar reconnaissance orbiter," *Journal of Geodesy*, vol. 86, pp. 193–207, 2012.

[49] C. Harris, M. Stephens, *et al.*, "A combined corner and edge detector," in *Alvey vision conference*, vol. 15, pp. 10–5244, Citeseer, 1988.

[50] T. Catuogno, E. E. Zini, C. Di Lauro, O. Trematerra, and C. Iacurto, "Crater matching for orbit determination boosted by neural networks,"

[51] J. J. Gibson, "The perception of the visual world.," 1950.

[52] C. S. Royden and K. D. Moore, "Use of speed cues in the detection of moving objects by moving observers," *Vision research*, vol. 59, pp. 17–24, 2012.

[53] K. R. Aires, A. M. Santana, and A. A. Medeiros, "Optical flow using color information: preliminary results," in *Proceedings of the 2008 ACM symposium on Applied computing*, pp. 1607–1611, 2008.

[54] G. L. Barrows, J. S. Chahl, and M. V. Srinivasan, "Biologically inspired visual sensing and flight control," *The Aeronautical Journal*, vol. 107, no. 1069, pp. 159–168, 2003.

[55] C. Brown and C. Brown, *Advances in Computer Vision: Volume 1*, vol. 1. Psychology Press, 2014.

[56] D. Fleet and Y. Weiss, "Optical flow estimation," in *Handbook of mathematical models in computer vision*, pp. 237–257, Springer, 2006.

[57] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques," *International journal of computer vision*, vol. 12, pp. 43–77, 1994.

[58] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *International journal of computer vision*, vol. 92, pp. 1–31, 2011.

[59] A. Reyes, A. Alba, and E. R. Arce-Santana, "Optical flow estimation using phase only-correlation," *Procedia Technology*, vol. 7, pp. 103–110, 2013.

[60] S. S. Sengar and S. Mukhopadhyay, "Motion detection using block based bi-directional optical flow method," *Journal of Visual Communication and Image Representation*, vol. 49, pp. 89–103, 2017.

[61] B. D. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision,” in *IJCAI’81: 7th international joint conference on Artificial intelligence*, vol. 2, pp. 674–679, 1981.

[62] B. K. Horn and B. G. Schunck, “Determining optical flow,” *Artificial intelligence*, vol. 17, no. 1-3, pp. 185–203, 1981.

[63] B. F. Buxton and H. Buxton, “Computation of optic flow from the motion of edge features in image sequences,” *Image and Vision Computing*, vol. 2, no. 2, pp. 59–75, 1984.

[64] A. Jepson and M. J. Black, “Mixture models for optical flow computation,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 760–761, IEEE, 1993.

[65] J. Weickert, A. Bruhn, N. Papenberg, and T. Brox, “Variational optic flow computation: From continuous models to algorithms,” *IWCVIA*, vol. 3, pp. 1–6, 2003.

[66] M. Menze, C. Heipke, and A. Geiger, “Discrete optimization for optical flow,” in *Pattern Recognition: 37th German Conference, GCPR 2015, Aachen, Germany, October 7-10, 2015, Proceedings 37*, pp. 16–28, Springer, 2015.

[67] G. Dantzig and D. R. Fulkerson, “On the max flow min cut theorem of networks,” *Linear inequalities and related systems*, vol. 38, pp. 225–231, 2003.

[68] G. Badura, C. Valenta, and B. Gunter, “Convolutional neural networks for inference of space object attitude status,” 09 2020.

[69] K. Thangavel, D. Spiller, R. Sabatini, S. Amici, S. T. Sasidharan, H. Fayek, and P. Marzocca, “Autonomous satellite wildfire detection using hyperspectral imagery and neural networks: A case study on australian wildfire,” *Remote Sensing*, vol. 15, no. 3, p. 720, 2023.

[70] V. S. Bilodeau, S. Clerc, R. Drai, and J. De Lafontaine, “Optical navigation system for pin-point lunar landing,” *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 10535–10542, 2014.

[71] A. Silburt, M. Ali-Dib, C. Zhu, A. Jackson, D. Valencia, Y. Kissin, D. Tamayo, and K. Menou, “Lunar crater identification via deep learning,” *Icarus*, vol. 317, pp. 27–38, 2019.

[72] O. Ronneberger, P. Fischer, and T. Brox, “Convolutional networks for biomedical image segmentation,” in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015 Conference Proceedings*, 2022.

[73] L. M. Downes, T. J. Steiner, and J. P. How, "Lunar terrain relative navigation using a convolutional neural network for visual crater detection," in *2020 American Control Conference (ACC)*, pp. 4448–4453, IEEE, 2020.

[74] L. Downes, T. J. Steiner, and J. P. How, "Deep learning crater detection for lunar terrain relative navigation," in *AIAA SciTech 2020 Forum*, p. 1838, 2020.

[75] D. M. DeLatte, S. T. Crites, N. Guttenberg, E. J. Tasker, and T. Yairi, "Segmentation convolutional neural networks for automatic crater detection on mars," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 8, pp. 2944–2957, 2019.

[76] E. Emami, T. Ahmad, G. Bebis, A. Nefian, and T. Fong, "Crater detection using unsupervised algorithms and convolutional neural networks," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 8, pp. 5373–5383, 2019.

[77] H. Yuen, J. Princen, J. Illingworth, and J. Kittler, "Comparative study of hough transform methods for circle finding," *Image and vision computing*, vol. 8, no. 1, pp. 71–77, 1990.

[78] D. W. Jacobs, "Robust and efficient detection of salient convex groups," *IEEE transactions on pattern analysis and machine intelligence*, vol. 18, no. 1, pp. 23–37, 1996.

[79] R. Alfredo *et al.*, "A robust crater matching algorithm for autonomous vision-based spacecraft navigation," in *2021 IEEE 8th International Workshop on Metrology for AeroSpace (MetroAeroSpace)*, pp. 322–327, IEEE, 2021.

[80] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[81] S. Silvestrini, M. Piccinin, G. Zanotti, A. Brandonisio, I. Bloise, L. Feruglio, P. Lunghi, M. Lavagna, and M. Varile, "Optical navigation for lunar landing based on convolutional neural network crater detector," *Aerospace Science and Technology*, vol. 123, p. 107503, 2022.

[82] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp. 234–241, Springer, 2015.

[83] J. Brownlee, “A gentle introduction to the rectified linear unit (relu),” *Machine learning mastery*, vol. 6, 2019.

[84] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pp. 21–37, Springer, 2016.

[85] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.

[86] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

[87] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, *et al.*, “Searching for mobilenetv3,” in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1314–1324, 2019.

[88] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, “Mnasnet: Platform-aware neural architecture search for mobile,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2820–2828, 2019.

[89] S. Elfwing, E. Uchibe, and K. Doya, “Sigmoid-weighted linear units for neural network function approximation in reinforcement learning,” *Neural networks*, vol. 107, pp. 3–11, 2018.

[90] P. Ramachandran, B. Zoph, and Q. V. Le, “Searching for activation functions,” *arXiv preprint arXiv:1710.05941*, 2017.

[91] D. Hendrycks and K. Gimpel, “Bridging nonlinearities and stochastic regularizers with gaussian error linear units,” *ArXiv*, vol. abs/1606.08415, 2016.

[92] Unibap, “Spacecloud iX5-100 datasheet,” 2015.

[93] M. Barker, E. Mazarico, G. Neumann, M. Zuber, J. Haruyama, and D. Smith, “A new lunar digital elevation model from the lunar orbiter laser altimeter and selene terrain camera,” *Icarus*, vol. 273, pp. 346–355, 2016.

[94] D. E. Smith, M. T. Zuber, G. A. Neumann, F. G. Lemoine, E. Mazarico, M. H. Torrence, J. F. McGarry, D. D. Rowlands, J. W. Head III, T. H. Duxbury, *et al.*, "Initial observations from the lunar orbiter laser altimeter (lola)," *Geophysical Research Letters*, vol. 37, no. 18, 2010.

[95] A. L. Shumway, M. Whiteley, J. Q. Peterson, Q. Young, and J. J. Hancock, "Digital imaging space camera (disc) design and testing," 2007.

[96] A. Wrona and A. Tantucci, "Artificial intelligence–based data path control in leo satellites–driven optical communications," *Authorea*, 2023.

[97] A. Yaqoob, T. Bi, and G.-M. Muntean, "A survey on adaptive 360 video streaming: Solutions, challenges and opportunities," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2801–2838, 2020.

[98] M. N. Sadiku, S. M. Musa, and O. D. Momoh, "Cloud computing: opportunities and challenges," *IEEE potentials*, vol. 33, no. 1, pp. 34–36, 2014.

[99] S. Li, L. D. Xu, and S. Zhao, "The internet of things: a survey," *Information systems frontiers*, vol. 17, pp. 243–259, 2015.

[100] H. Kaushal, V. Jain, and S. Kar, *Free space optical communication.* Springer, 2017.

[101] D. W. Ball, "The electromagnetic spectrum: a history," *Spectroscopy*, vol. 22, no. 3, p. 14, 2007.

[102] D. Killinger, "Free space optics for laser communication through the air," *Optics and photonics news*, vol. 13, no. 10, pp. 36–42, 2002.

[103] H. Burchardt, N. Serafimovski, D. Tsonev, S. Videv, and H. Haas, "Vlc: Beyond point-to-point communication," *IEEE Communications Magazine*, vol. 52, no. 7, pp. 98–105, 2014.

[104] A. Biswas, M. Srinivasan, S. Piazzolla, and D. Hoppe, "Deep space optical communications," in *Free-Space Laser Communication and Atmospheric Propagation XXX*, vol. 10524, pp. 242–252, SPIE, 2018.

[105] H. Hauschildt, C. Elia, H. L. Moeller, and D. Schmitt, "Scylight—esa's secure and laser communication technology framework for satcom," in *2017 IEEE International Conference on Space Optical Systems and Applications (ICSOS)*, pp. 250–254, IEEE, 2017.

[106] L. C. Andrews, R. L. Phillips, C. Y. Hopen, and M. Al-Habash, "Theory of optical scintillation," *JOSA A*, vol. 16, no. 6, pp. 1417–1429, 1999.

[107] G. Berman, A. Chumak, and V. Gorshkov, “Beam wandering in the atmosphere: The effect of partial coherence,” *Physical Review E*, vol. 76, no. 5, p. 056606, 2007.

[108] Z. Wang, J. Zhang, and H. Gao, “Impacts of laser beam divergence on lidar multiple scattering polarization returns from water clouds,” *Journal of Quantitative Spectroscopy and Radiative Transfer*, vol. 268, p. 107618, 2021.

[109] P. Chitre and F. Yegenoglu, “Next-generation satellite networks: architectures and implementations,” *IEEE Communications Magazine*, vol. 37, no. 3, pp. 30–36, 1999.

[110] X. Mao, D. Arnold, V. Girardin, A. Villiger, and A. Jäggi, “Dynamic gps-based leo orbit determination with 1 cm precision using the bernese gnss software,” *Advances in space research*, vol. 67, no. 2, pp. 788–805, 2021.

[111] C. B. Lim, A. Montmerle-Bonnefois, C. Petit, J.-F. Sauvage, S. Meimon, P. Perrault, F. Mendez, B. Fleury, J. Montri, J.-M. Conan, *et al.*, “Single-mode fiber coupling with adaptive optics for free-space optical communication under strong scintillation,” in *2019 IEEE International Conference on Space Optical Systems and Applications (ICSOS)*, pp. 1–6, IEEE, 2019.

[112] C. Zhang, J. Jin, L. Kuang, and J. Yan, “Leo constellation design methodology for observing multi-targets,” *Astrodynamics*, vol. 2, pp. 121–131, 2018.

[113] Y. Su, Y. Liu, Y. Zhou, J. Yuan, H. Cao, and J. Shi, “Broadband leo satellite communications: Architectures and key technologies,” *IEEE Wireless Communications*, vol. 26, no. 2, pp. 55–61, 2019.

[114] P. Yue, J. An, J. Zhang, J. Ye, G. Pan, S. Wang, P. Xiao, and L. Hanzo, “Low earth orbit satellite security and reliability: Issues, solutions, and the road ahead,” *IEEE Communications Surveys & Tutorials*, 2023.

[115] P. K. Chowdhury, M. Atiquzzaman, and W. Ivancic, “Handover schemes in satellite networks: State-of-the-art and future research directions,” *IEEE Communications Surveys & Tutorials*, vol. 8, no. 4, pp. 2–14, 2006.

[116] S. He, T. Wang, and S. Wang, “Load-aware satellite handover strategy based on multi-agent reinforcement learning,” in *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pp. 1–6, IEEE, 2020.

[117] M. Kasper, E. Fedrigo, D. P. Looze, H. Bonnet, L. Ivanescu, and S. Oberti, “Fast calibration of high-order adaptive optics systems,” *JOSA A*, vol. 21, no. 6, pp. 1004–1008, 2004.

[118] R. K. Tyson and B. W. Frazier, *Principles of adaptive optics.* CRC press, 2022.

[119] G. Vosselman and H.-g. Maas, "Adjustment and filtering of raw laser altimetry data," in *Proceedings of OEEPE Workshop on Airborne Laserscanning and Interferometric SAR for Detailed Digital Terrain Models, Stockholm, Sweden*, OEEPE, 2001.

[120] S. W. Jolly, O. Gobert, and F. Quéré, "Spatio-temporal characterization of ultrashort laser beams: a tutorial," *Journal of Optics*, vol. 22, no. 10, p. 103501, 2020.

[121] A. Wrona, E. De Santis, F. D. Priscoli, and F. G. Lavacca, "An intelligent ground station selection algorithm in satellite optical communications via deep learning," in *2023 31st Mediterranean Conference on Control and Automation (MED)*, pp. 493–499, IEEE, 2023.

[122] B. Rödiger, D. Ginthör, J. P. Labrador, J. Ramirez, C. Schmidt, and C. Fuchs, "Demonstration of an fso/rf hybrid-communication system on aeronautical and space applications," in *Laser Communication and Propagation through the Atmosphere and Oceans IX*, vol. 11506, p. 1150603, SPIE, 2020.

[123] C. Sinka and J. Bitó, "Site diversity against rain fading in lmds systems," *IEEE microwave and wireless components letters*, vol. 13, no. 8, pp. 317–319, 2003.

[124] T. Nakatani, Y. Maekawa, Y. Shibagaki, and K. Hatsuda, "Relationship between rain front motion and site diversity in ku-band satellite links," in *25th AIAA International Communications Satellite Systems Conference (organized by APSCC)*, p. 3173, APSCC, 2007.

[125] P. Petropoulou, E. T. Michailidis, A. D. Panagopoulos, and A. G. Kanatas, "Radio propagation channel measurements for multi-antenna satellite communication systems: A survey," *IEEE Antennas and Propagation Magazine*, vol. 56, no. 6, pp. 102–122, 2014.

[126] J. P. Baptista and P. Davies, "Reference book on attenuation measurement and prediction," in *2nd Workshop OPEX*, OPEX, 1994.

[127] J. Goldhirsh, B. H. Musiani, A. W. Dissanayake, and K.-T. Lin, "Three-site space-diversity experiment at 20 ghz using acts in the eastern united states," *Proceedings of the IEEE*, vol. 85, no. 6, pp. 970–980, 1997.

[128] S. Lin, H. Bergmann, and M. Pursley, “Rain attenuation on earth-satellite paths—summary of 10-year experiments and studies,” *Bell System Technical Journal*, vol. 59, no. 2, pp. 183–228, 1980.

[129] M. Luglio, R. Mancini, C. Riva, A. Paraboni, and F. Barbaliscia, “Large-scale site diversity for satellite communication networks,” *International journal of satellite communications*, vol. 20, no. 4, pp. 251–260, 2002.

[130] C. Bruni, F. D. Priscoli, G. Koch, A. Pietrabissa, and L. Pimpinella, “Network decomposition and multi-path routing optimal control,” *Transactions on Emerging Telecommunications Technologies*, vol. 24, pp. 154–165, May 2012.

[131] A. Pietrabissa, L. R. Celsi, F. Cimorelli, V. Suraci, F. D. Priscoli, A. D. Giorgio, A. Giuseppi, and S. Monaco, “Lyapunov-based design of a distributed wardrop load-balancing algorithm with application to software-defined networking,” *IEEE Transactions on Control Systems Technology*, vol. 27, pp. 1924–1936, Sept. 2019.

[132] S. Poulenard, M. Crosnier, and A. Rissons, “Ground segment design for broadband geostationary satellite with optical feeder link,” *Journal of Optical Communications and Networking*, vol. 7, no. 4, pp. 325–336, 2015.

[133] T. Rossi, M. De Sanctis, F. Maggio, M. Ruggieri, C. Hibberd, and C. Togni, “Smart gateway diversity optimization for ehf satellite networks,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 1, pp. 130–141, 2019.

[134] C. N. Efrem and A. D. Panagopoulos, “Globally optimal selection of ground stations in satellite systems with site diversity,” *IEEE Wireless Communications Letters*, vol. 9, no. 7, pp. 1101–1104, 2020.

[135] C. Fuchs and F. Moll, “Ground station network optimization for space-to-ground optical communication links,” *Journal of Optical Communications and Networking*, vol. 7, no. 12, pp. 1148–1159, 2015.

[136] N. K. Lyras, C. N. Efrem, C. I. Kourogiorgas, and A. D. Panagopoulos, “Optimum monthly based selection of ground stations for optical satellite networks,” *IEEE Communications Letters*, vol. 22, no. 6, pp. 1192–1195, 2018.

[137] E. Erdogan, I. Altunbas, G. K. Kurt, M. Bellemare, G. Lamontagne, and H. Yanikomeroglu, “Site diversity in downlink optical satellite networks through ground station selection,” *IEEE Access*, vol. 9, pp. 31179–31190, 2021.

[138] X. Hu, Y. Zhang, X. Liao, Z. Liu, W. Wang, and F. M. Ghannouchi, "Dynamic beam hopping method based on multi-objective deep reinforcement learning for next generation satellite broadband systems," *IEEE Transactions on Broadcasting*, vol. 66, no. 3, pp. 630–646, 2020.

[139] X. Hu, S. Liu, Y. Wang, L. Xu, Y. Zhang, C. Wang, and W. Wang, "Deep reinforcement learning-based beam hopping algorithm in multibeam satellite systems," *IET Communications*, vol. 13, no. 16, pp. 2485–2491, 2019.

[140] S. Liu, X. Hu, and W. Wang, "Deep reinforcement learning based dynamic channel allocation algorithm in multibeam satellite systems," *IEEE Access*, vol. 6, pp. 15733–15742, 2018.

[141] X. Hu, S. Liu, R. Chen, W. Wang, and C. Wang, "A deep reinforcement learning-based framework for dynamic resource allocation in multibeam satellite systems," *IEEE Communications Letters*, vol. 22, no. 8, pp. 1612–1615, 2018.

[142] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[143] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[144] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018.

[145] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[146] N. Ashby, "The sagnac effect in the global positioning system," in *Relativity in rotating frames: relativistic physics in rotating reference frames*, pp. 11–28, Springer, 2004.

[147] D. Beniaguev, "Historical hourly weather data 2012-2017." `https://www.kaggle.com/datasets/selfishgene/historical-hourly-weather-data`, Dec 2017. Accessed June 28, 2023.

[148] GMT, "Astronomical tables for equinoxes and solstices." `https://greenwichmeantime.com/longest-day/equinox-solstice-2010-2020/`. Accessed July 6, 2023.

[149] H. Henniger and O. Wilfert, "An introduction to free-space optical communications.," *Radioengineering*, vol. 19, no. 2, 2010.

[150] J. San-Miguel-Ayanz, T. Durrant, R. Boca, P. Maianti, G. Liberta', D. Oom, A. Branco, D. De Rigo, D. Ferrari, E. Roglia, and N. Scionti, "Advance report on forest fires in europe, middle east and north africa 2022," no. KJ-NA-31-479-EN-N (online), 2023.

[151] J. San-Miguel-Ayanz, R. Boca, T. Durrant, P. Maianti, G. Liberta', D. Oom, A. Branco, D. De Rigo, D. Ferrari, E. Roglia, and N. Scionti, "Forest fires in europe, middle east and north africa 2021," no. KJ-NA-31-269-EN-N (online),KJ-NA-31-269-EN-C (print), 2022.

[152] J. San-Miguel-Ayanz, T. Durrant, R. Boca, P. Maianti, G. Liberta', T. Artes Vivancos, D. Jacome Felix Oom, A. Branco, D. De Rigo, D. Ferrari, H. Pfeiffer, R. Grecchi, D. Nuijten, and T. Leray, "Forest fires in europe, middle east and north africa 2020," no. KJ-NA-30862-EN-N (online) , KJ-NA-30862-EN-C (print), 2021.

[153] J. San-Miguel-Ayanz, R. Boca, T. Durrant, P. Maianti, G. Liberta', T. Artes Vivancos, D. Jacome Felix Oom, A. Branco, D. De Rigo, D. Ferrari, H. Pfeiffer, R. Grecchi, D. Nuijten, and T. Leray, "Forest fires in europe, middle east and north africa 2019," no. KJ-NA-30402-EN-N (online),KJ-NA-30402-EN-C (print), 2020.

[154] J. San-Miguel-Ayanz, T. Durrant, R. Boca, G. Liberta', A. Branco, D. De Rigo, D. Ferrari, P. Maianti, T. Artes Vivancos, H. Pfeiffer, P. Loffler, D. Nuijten, T. Leray, and D. Jacome Felix Oom, "Forest fires in europe, middle east and north africa 2018," no. KJ-NA-29856-EN-N (online),KJ-NA-29856-EN-C (print), 2019.

[155] K. Yamaguchi, K. Sakamoto, T. Akabane, and Y. Fujimoto, "A neural network for speaker-independent isolated word recognition.," in *ICSLP*, 1990.

[156] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[157] I. Goodfellow, Y. Bengio, and A. Courville, "Softmax units for multinoulli output distributions. deep learning," 2018.

[158] R. G. Gottlieb, "Fast gravity, gravity partials, normalized gravity, gravity gradient torque and magnetic field: derivation, code and data," tech. rep., 1993.

[159] J. Cappellari, C. Vélez, and A. J. Fuchs, *Mathematical theory of the Goddard trajectory determination system*, vol. 71106. Goddard Space Flight Center, 1976.