ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Aerospace Engineering (29th Cycle)

Artificial Intelligence for Small Satellites Mission Autonomy

By

Lorenzo Feruglio

******

Supervisor:
Prof. S. Corpino

Doctoral Examination Committee:
Prof. Franco Bernelli Zazzera, Referee, Politecnico di Milano
Prof. Michèle Roberta Jans Lavagna, Referee, Politecnico di Milano
Prof. Giancarmine Fasano, Referee, Università di Napoli Federico II
Prof. Paolo Maggiore, Referee, Politecnico di Torino
Prof. Nicole Viola, Referee, Politecnico di Torino

Politecnico di Torino
2017

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Lorenzo Feruglio

2017

* This dissertation is presented in partial fulfilment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

A mia mamma, mio papà, mio fratello.

Grazie per esserci sempre stati,

per essere stati delle guide incredibili.

Grazie

Acknowledgment

I would like to acknowledge and thank a great number of people: not everyone can be included here, but I'm sure the people I would like to thank already know I'm grateful to them.

Valentina, the road will be long, but you already gave me the spark to start what I've always wanted to do. Thank you my little strawberry.

My supervisor, Sabrina, for giving me the freedom to explore, even if the navigation was in uncharted territories. That has been a nice roaming, and was only the prelude to something bigger!

My team, in general, for sharing with me years (from 2009!!) of successful and unsuccessful trials and errors, and motivating research.

Raffaele, Fabio, Gerard, Fabrizio, for spending time on a bunch of electronics boards for so many years: it was worth the ride!

My mentor in the US, Alessandra, for giving me the chance to visit a wonderful research centre, and for the time spent working together there.

The two satellites that flew in orbit, one in 2012 (e-st@r-I) and the other one in 2016 (e-st@r-II) and is still there: you give me huge chances to brag about the amazing things I've had the joy to be part of. Eh, my codes have been in orbit two times already!

Loris and Giorgio for starting the new adventure together and believing in this.

Loris, Martina, Raffaele, Daniele, Christian, Christopher and Pietro for making the workplace a lovely place to spend time at. I haaaaaaaaaaaaloveaaaaate you guys ;)

At last, my other dear friends: Stefano, Francesco, Paolo, Gabriele: I've shared so much with you and I have no intention of stopping.

Abstract

Space mission engineering has always been recognized as a very challenging and innovative branch of engineering: since the beginning of the space race, numerous milestones, key successes and failures, improvements, and connections with other engineering domains have been reached. Despite its relative young age, space engineering discipline has not gone through homogeneous times: alternation of leading nations, shifts in public and private interests, allocations of resources to different domains and goals are all examples of an intrinsic dynamism that characterized this discipline. The dynamism is even more striking in the last two decades, in which several factors contributed to the fervour of this period. Two of the most important ones were certainly the increased presence and push of the commercial and private sector and the overall intent of reducing the size of the spacecraft while maintaining comparable level of performances. A key example of the second driver is the introduction, in 1999, of a new category of space systems called CubeSats. Envisioned and designed to ease the access to space for universities, by standardizing the development of the spacecraft and by ensuring high probabilities of acceptance as piggyback customers in launches, the standard was quickly adopted not only by universities, but also by agencies and private companies. CubeSats turned out to be a disruptive innovation, and the space mission ecosystem was deeply changed by this. New mission concepts and architectures are being developed: CubeSats are now considered as secondary payloads of bigger missions, constellations are being deployed in Low Earth Orbit to perform observation missions to a performance level considered to be only achievable by traditional, fully-sized spacecraft.

CubeSats, and more in general the small satellites technology, had to overcome important challenges in the last few years that were constraining and reducing the diffusion and adoption potential of smaller spacecraft for scientific and technology demonstration missions. Among these challenges were: the miniaturization of propulsion technologies, to enable concepts such as Rendezvous and Docking, or interplanetary missions; the improvement of telecommunication state of the art for small satellites, to enable the downlink to Earth of all the data acquired during the mission; and the miniaturization of scientific instruments, to be able to exploit CubeSats in more meaningful, scientific, ways. With the size reduction and with the consolidation of the technology, many aspects of a space mission are reduced in consequence: among these, costs, development and launch times can be cited. An important aspect that has not been demonstrated to scale accordingly is operations: even for small satellite missions, human operators and performant ground control centres are needed. In addition, with the possibility of having constellations or interplanetary distributed missions, a redesign of how operations are management is required, to cope with the innovation in space mission architectures.

The present work has been carried out to address the issue of operations for small satellite missions. The thesis presents a research, carried out in several institutions (Politecnico di Torino, MIT, NASA JPL), aimed at improving the autonomy level of space missions, and in particular of small satellites. The key technology exploited in the research is Artificial Intelligence, a computer science branch that has gained extreme interest in research disciplines such as medicine, security, image recognition and language processing, and is currently making its way in space engineering as well. The thesis focuses on three topics, and three related applications have been developed and are here presented: autonomous operations by means of event detection algorithms, intelligent failure detection on small satellite actuator systems, and decision-making support thanks to intelligent tradespace exploration during the preliminary design of space missions. The

Artificial Intelligent technologies explored are: Machine Learning, and in particular Neural Networks; Knowledge-based Systems, and in particular Fuzzy Logics; Evolutionary Algorithms, and in particular Genetic Algorithms. The thesis covers the domain (small satellites), the technology (Artificial Intelligence), the focus (mission autonomy) and presents three case studies, that demonstrate the feasibility of employing Artificial Intelligence to enhance how missions are currently operated and designed.

Contents

List of Figures

List of Tables

Notation

ADC	–	Analog-to-Digital Converter
AI	–	Artificial Intelligence
ANN	–	Artificial Neural Networks
AOCS	–	Attitude and Orbit Control System
C&DH	–	Command and Data Handling
CDS	–	CubeSat Design Specification
COTS	–	Components Off The Shelf
CS	–	Computer Science
DL	–	Deep Learning
EA	–	Evolutionary Algorithm
ED	–	Event Detection
EDL	–	Entry, Descent and Landing
EM-1	–	Exploration Mission 1
EMF	–	Earth Magnetic Field
EO	–	Earth Observation
EP	–	Evolutionary Programming
EPS	–	Electrical Power System
ES	–	Expert System (used in 5.7 and in 7)
ES	–	Evolution Strategies (used in 5.8)
ESA	–	European Space Agency
FDIR	–	Failure Detection, Isolation and Recovery
FL	–	Fuzzy Logics
FSW	–	Flight Software
GA	–	Genetic Algorithm
GCS	–	Ground Control Station
GNC	–	Guidance, Navigation and Control
GP	–	Genetic Programming
GPU	–	Graphics Processing Unit
GS	–	Ground Segment
GSTP	–	General Support Technology Programme
HEAO	–	High Energy Astronomical Observatory
HL	–	Hidden Layer
HO	–	Hard-Over
HST	–	Hubble Space Telescope
HW	–	Hardware
IE	–	Inference Engine
IOD	–	In Orbit Demonstration
ISS	–	International Space Station

J-SSOD	–	Japanese Experiment Module Small Sat Orbital Deployer
JEMRMS	–	Japanese Experiment Module Remote Manipulator System
KB	–	Knowledge Base
KBE	–	Knowledge-based Engineering
KBS	–	Knowledge-based System
LEO	–	Low Earth Orbit
LIP	–	Lock In Place
LOE	–	Loss Of Efficiency
MATE	–	Multi Attribute Tradespace Exploration
MAUT	–	Multi Attribute Utility Theory
MC	–	Mission Control
MDP	–	Markov Decision Processes
ML	–	Machine Learning
MT	–	Magnetic Torquer
NASA	–	National Aeronautics and Space Administration
NEA	–	Near Earth Asteroid
NN	–	Neural Network
NRCSD	–	NanoRacks CubeSat Deployer
OEM	–	Original Equipment Manufacturer
OS	–	Operating Systems
P&S	–	Planning and Scheduling
P-POD	–	Poly-Picosatellite Orbital Deployer
PAC-L	–	Probably Approximately Correct Learning
RAM	–	Random Access Memory
ROM	–	Read Only Memory
RT	–	Real Time
RTOS	–	Real Time Operating System
SCG	–	Scaled Conjugate Gradient
SI	–	Science Instrument
SLS	–	Space Launch System
SMM	–	Solar Maximum Mission
SoA	–	State of the Art
SS	–	Space Segment
SW	–	Software
UAV	–	Unmanned Aerial Vehicle

Chapter 1

Introduction

The last two decades have been interesting times for space missions and have seen a dedicated effort among the major players in the space domain to design and develop unmanned mission ideas and concepts that are more challenging than ever. Thanks to the consistent successes of great interplanetary and Earth orbiting missions, space engineering has been pushing the boundaries for constant improvement, envisioning everyday increasingly daring missions. The traditional, monolithic, high-performance spacecraft have not been the only category of space systems influenced by this push in innovation and in ambition: smaller satellites have been gaining traction, thanks to newly developed technologies and to a consolidation of the present state of the art. Small satellites, nanosatellites, CubeSats, are experiencing a renovated and never-before-seen interest and exploitation, thanks to the game-changing characteristics that this type of space systems possess. The effort in using smaller satellites is common and shared among the major agencies and industries in the world panorama.

Since 2013, ESA has initiated seven different CubeSat projects for low-cost In-Orbit Demonstration (IOD) of innovative miniaturized technologies within the framework of Element 3 of the General Support Technology Programme (GSTP). The first technology IOD CubeSat to be launched, a 3U CubeSat called GOMX-3, was deployed from ISS in October 2015 and has been a complete success over its 1-year lifetime in Low Earth Orbit (LEO) until re-entry. Other IOD CubeSats in development are planned for launch in 2017 and 2018. Additional design effort has been spent at ESA to study the applicability of small satellites for interplanetary or

lunar missions. These concepts are mostly based on mother-daughter architectures where the mothercraft transports the CubeSats to a target destination, deploys them locally to perform their mission, and provides data relay support back to Earth for TT/C and payload data downlink, enabled by a bi-directional inter-satellite link.

NASA has been following a similar approach, by studying the potential exploitation of small satellites, for supporting flagship missions in the Solar System. Moreover, concepts based on the CubeSat technology have been appearing even for planet-based missions, such as the Mars Helicopter concept or a Europa under-ice explorer.

Moreover, with the ongoing development of miniaturized solar array drive assemblies for relatively high power steerable solar arrays, high delta-V Electric Propulsion subsystems, and deep space X-band transponders with high gain antenna reflectarrays, stand-alone interplanetary CubeSat missions are also being considered, based on 12U CubeSat form factor and exploitation of piggyback launch opportunities to near-Earth escape, thus opening up the potential for truly low-cost space exploration.

Thanks to the efforts in technology miniaturization, thanks to the appearance of radiation-hardened COTS and tighter system integration, significant reductions in space and launch segment costs of entry-level spacecraft are enabled. Unfortunately, the operations costs do not scale down with spacecraft size/mass. For certain kinds of missions, especially in a mother/daughter architecture, the CubeSat can reach complexity levels comparable to those of the mothership, if classical operational approaches are used. Moreover, due to limitations in the telecommunication windows and timings of interplanetary missions, Earth-based control and monitoring may be infrequent for small spacecraft. Limitations in the data rate available (constrained most of the times by the distances, the system sizes involved, and the available power on board) and the consequent costs of more performant ground systems (to overcome the lower onboard performances), add further complexity and limitations to a typical small satellite mission. It becomes evident that, to achieve truly low-cost ambitious small satellite missions, a high degree of onboard autonomy will be required to ensure the missions is executed despite limited ground contact and with a reduced mission operations centre. Finally, aiming at innovating and improving the operations architecture is a must when considering constellation missions composed by tens or hundreds of small satellites.

Thanks to the faster development cycles of COTS components and weaker quality- and reliability-oriented approaches, small satellites are often employing high computational capabilities within low power consumption and small form factors. This enables advanced and computationally-intensive autonomy approaches to be run onboard, compared to larger missions.

1.1 Thesis Objectives

The objective of the research presented in this thesis is the following:

> ***Exploring*** *the role and capabilities of* ***Artificial Intelligence based algorithms****, to significantly* ***increase the mission and system autonomy of Small Satellite missions,*** *investigating the* ***feasibility*** *of using these algorithms by implementing and testing* ***working prototypes****.*

To this purpose, Artificial Intelligence approaches and algorithms can be implemented into space missions with the objective of enhancing the autonomous decision-making capabilities of the space segment in terms of:

- Emulation of the expert knowledge required for mission operations
- Execution of tasks that cannot be defined during the development of the spacecraft
- Optimization of onboard resources and execution of specific tasks

thereby ultimately leading to a reduction in operations costs for future small missions through smaller operations teams and less frequent usage of large, deep space, ground station network antennas. The presented research focused on identification and application of Artificial Intelligence algorithms to enable smart payload operations planning, fault detection, and, targeting the preliminary design phase of a mission, intelligent spacecraft design.

This being said, it is important to consider that the thesis developed is presented as a conclusion of an Aerospace program: the thesis and the research work performed did not have the objective of determining which, among the available Artificial Intelligence algorithms, is the best candidate to perform the automation of a certain type of operations. Instead, the research is meant to be considered as a feasibility study for developing AI-based solutions to real operations problems. Additional studies and comparisons will have to follow in order to assess whether the proposed algorithms are in fact the best options to solve the problem addressed

in the case studies. Moreover, chosen candidates will have to be compared in future works with other, non-AI-based algorithms.

The target spacecraft platform used in the thesis is constituted by a group of heterogenous spacecraft categories that are commonly known as Small Satellites, Nanosatellites, CubeSats and so on. Despite these category labels carry very precise meaning and represent distinct typologies of space systems, for the purpose of easiness of reading, and given the fact that no substantial change happens when switching among the aforementioned categories when dealing with mission autonomy, the following statement holds true throughout the whole thesis:

> *Small Satellite, Nanosatellite, CubeSats, Microsatellites and other similar terms are used interchangeably and identify a common category of spacecraft that encompasses several accepted categories, provided that all of them used refer to spacecraft of limited mass and dimension, and characterized by substantially different architectures and features with respect to traditional missions.*

1.2 Thesis layout

The thesis follows a straightforward layout, presenting the category of space systems that serves as basis for the work, the domain (the software) that is object of improvement, the functionalities to be implemented (autonomous operations) and the technology that enables these improvements (Artificial Intelligence). Finally, case studies demonstrate the findings of the research.

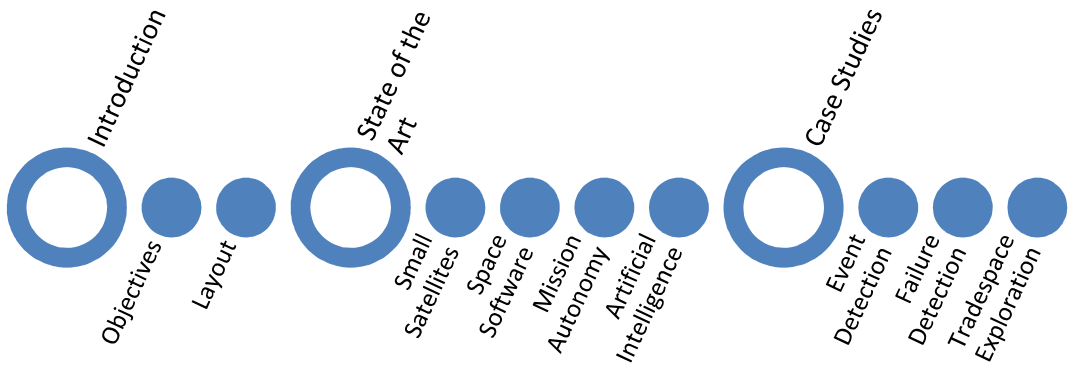


Figure 1 Thesis structure. Bigger circle represents the main conceptual sections of the thesis.

Chapter 2

Chapter 2 provides an overview of Small Satellites, and presents an historical overview of the most important Small Satellite missions. Moreover, a sub-category of the Small Satellites is presented, that acquired significant industrial interest in the last decades: CubeSats. Of this type of standardized spacecraft, the most important details are covered: the standard, the deployer technologies, and a market and diffusion analysis are presented. Finally, Chapter 2 presents some examples of the most striking and interesting Small Satellites and CubeSat missions over the years.

Chapter 3

Chapter 3 is about software, both ground-based and flight software. Overview of the most common approaches and functionalities present in space software are presented. The subject of space software is certainly vast: the presented concepts serve as a summarization of the different aspect to be considered during the design of space software. The chapter is not meant to include every possible aspect of software design approaches.

Chapter 4

Chapter 4 is the first of the two major chapters of this thesis, and introduces the concept of Mission Autonomy. The Chapter discusses about the need of improving Mission Autonomy on modern spacecraft, presents key terminology used throughout the thesis and discusses about past practices and current standards of autonomous operations on spacecraft. Finally, it presents the various issues that are currently driving the development on Mission Autonomy: control and operation management of big constellations, interplanetary missions performed with Small Satellites and unreliable ground support.

Chapter 5

The focus of Chapter 5 is on Artificial Intelligence. When dealing with these innovative algorithms in a new context, it is important to cover history and characteristics of the most dominant algorithms developed so far, even if not yet adapted for space applications. Chapter 5 defines Artificial Intelligence as a concept, and defines the State of the Art for this technology, from three different perspectives: by Algorithm (discussing the various algorithms that populate the domain of Artificial Intelligence), by application (presenting particular cases in which Artificial Intelligence plays a relevant role), and by open-source products

(listing the open-source technologies, frameworks and software that can be used to develop Artificial Intelligence applications, both for space or other domains). The chapter then focuses on three category of algorithms that were used in the case studies of the thesis: Machine Learning, and in particular Neural Networks, Expert Systems, and in particular Fuzzy Logics, and finally Evolutionary Algorithms, in particular Genetic Algorithms.

Chapter 6, 7 and 8: The Case Studies

Chapter 6, 7 and 8 present three case studies developed for this research: respectively Event Detection, Failure Detection and Tradespace Exploration. The Event Detection case is developed using Neural Networks: an algorithm and an innovative training approach is presented to be used during interplanetary missions on a comet / asteroid, enabling detection of impact events or spontaneous gas emissions. The Failure Detection case presents the use of Expert Systems to detect failures that happen on a common actuator of a Small Satellite, Magnetic Torquers. The presented approach performs considerably well on this category of components, but is at the same time easily re-configurable to work on other types of actuators or sensors of a spacecraft. Finally, the Tradespace Exploration case presents the use of Genetic Algorithms exploited to support decision makers (in this application, mission designers) in performing a very fast analysis on all the possible alternate solutions for the design of a specific mission.

Chapter 2

Small Satellites

2.1 Small Satellites and smaller systems

"Small Satellites" is term that defines a category of space systems, in particular of satellites. Although the term is not standardized and different interpretation of it exist, it is traditionally associated with systems of limited dimensions and mass inferior to 1000 kilograms.

Table 1 Small Satellites and related categories

Space agencies	Classification	Mass [kg]
European Space Agency (ESA) [1]	Small	350 - 700
	Mini	80 - 350
	Micro	50 - 80
Airbus Defence and Space [1]	miniXL	1000 - 1300
	Mini	400 - 700
	Micro	100 - 200
National Aeronautics and Space Administration (NASA) [2]	Minisatellite	100 - 180
	Microsatellite	100 - 100
	Nanosatellite (CubeSat)	1 - 10
	Femto- and Picosatellite	< 1

Most widely accepted [3]	SmallSat	500 - 1000
	MiniSat	100 - 500
	MicroSat	10 - 100
	NanoSat	1 - 10
	PicoSat	0.1 - 1
	FemtoSat	< 0.1

Refer to 1.1 for the interpretation of “Small Satellite” throughout the presented research. Different entities (being them space agencies or companies) implement their own classification based on satellite dimension, and most of them overlap, as summarized in Table 1 [4].

Despite the lack of fully standardized classification, the majority of the entities in the space industry agree on common aspects:

- Substantial changes in the architecture, design and implementation of satellites take place when the mass involved is less than 1000 kg
- When distinguishing the various types of satellites, the mass classification is one of the most useful [4]

From an historical point of view, from the launch of the first satellite (the Sputnik-1, launched in 1957 with a mass of 84 kg) the size trend of satellites has moved towards bigger, more complex, redundant and better performing systems. This trend has been evident in several categories of satellites, from Earth observation ones to geostationary telecommunication satellites. With the advent of small satellites, and in particular of nano-satellites, the proportion between the different categories of launched systems have shifted considerably. Market predictions for nano- and micro-satellite launches show a sustained growth in the number of satellites launched (Figure 2). Nonetheless, the small satellite trend is clear and showing defined growth.

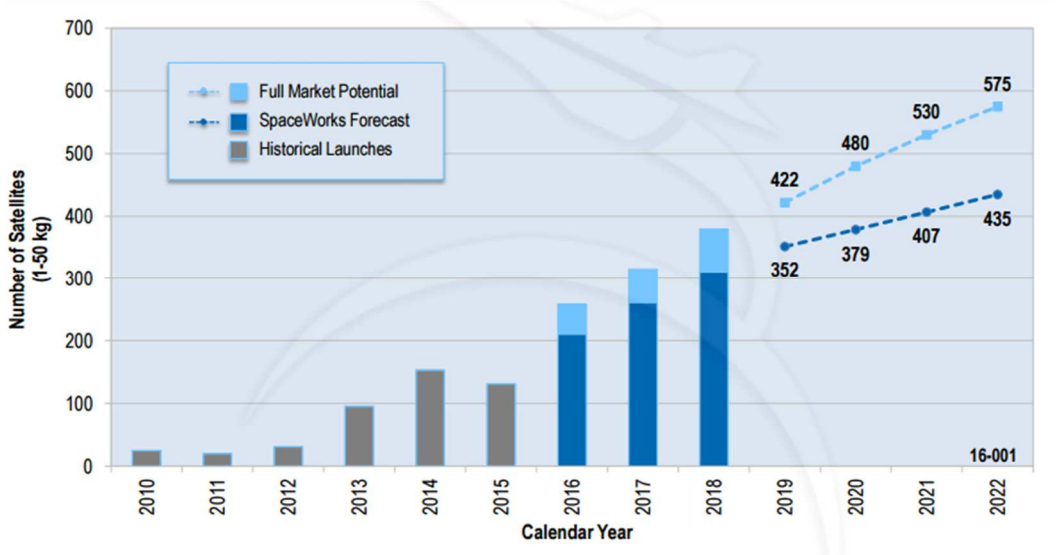


Figure 2 Nano- and Microsatellite launch history and forecast at 2015 (1 - 50 kg) – Credits SpaceWorks®

The evolution and diffusion of small satellites as major actor in the field of space missions have been possible also thanks to the improvements and advancements in electrical and mechanical miniaturization, that made possible the development of payloads and platforms that perform in a similar way to their bigger counterparts found in traditional assets. Antennas, cameras, spectrometers and so on, are example of the quality (and reduced sizes) reached in the last decades by these complex technologies [5]–[8]. Another important factor that led to the adoption of the small satellite category worldwide, is the great success this technology obtained in the educational sector. Thanks to the much more affordable costs and more agile development times and approaches, small satellites programs, teams and mission have begun to appear in different institutions: ESA ([9], [10]), NASA ([11]–[13]) among agencies, and several universities (Politecnico di Torino [14], University of Montpellier-2 [15], and more [16]).

2.2 CubeSats

Categorizing satellites by mass is not the only way, as other means (such as mission objectives, launch orbits and so on) could re-arrange the satellite database in other, still meaningful ways. Often times, categorizing satellite systems in different ways produces overlapping representations of the satellite missions ecosystem. A well-known example of this phenomenon is constituted by CubeSats (Figure 3).

Figure 3 CubeSat spacecraft. The three winners of first ESA Fly Your Satellite! competition: OUFTI-1, e-st@r-II, AAUSAT-4. Credits ESA

CubeSats are a category of space systems developed according to an open-source standard, proposed for the first time in 1999 by professors Jordi Puig-Suari of California Polytechnic State University and Bob Twiggs of Stanford University [17]. The objective behind the definition of the standard was to create a spacecraft system concept that would not only allow university groups to rapidly design and develop a small space project, but also would ensure that the chances of being accepted on traditional launchers as a secondary payload were maximised. To reach stable rates of acceptance among launch providers, the standard was designed to cover not only the space system itself, but also its interfaces with the launcher, via the design of a deployment system able to guarantee safeness for the other, most of the times more expensive and demanding, spacecraft on the launcher. In the initial vision, the CubeSat development would require less than 100.000$ to build for each One Unit (1U), allowing in addition a short duration of the launch procurement phase. In general, time and cost of the development can vary significantly depending on several factors, among which institution carrying out the project, budget and quality level are the most influencing ones. As introduced above, the CubeSat spacecraft encompass different size and mass categories, starting from the nanosatellite one to the microsatellite one.

2.2.1 Overview

The CubeSat platform is envisioned as a miniaturised satellite based on a standardized unit of mass and volume. A CubeSat spacecraft has the following characteristics in its base form, that is the 1U configuration:

- Dimension of 10 x 10 x 10 cm
- Mass up to 1.33 kg (originally 1 kg until 2009)
- Modularity
- Standardized requirements

Furthermore, the standard foresees additional spacecraft, with increasing sizes, in the factors of 1.5U, 2U, 3U, 4U, 6U, 8U and 12U.

2.2.2 The Standard

The CubeSat standard defines several characteristics of this category of space systems [17]:

- Interfaces
- Requirements (General, mechanical, electrical, operational, testing)
- Tolerances and dimensions
- Waiver forms and acceptance checklists
- Deployer characteristics

These characteristics are peculiar, and tend to be rigorously applied for each spacecraft in the category. In some cases, depending on the market availability of the deployers, some parameters are revised for each spacecraft, reducing the standardization of the CubeSats.

In general, it is possible to highlight some interesting features and requirements dictated by the CubeSat Design Specification.

Table 2 Extract of interesting CubeSat requirements from CDS rev. 13

Req. N.	Category	Description
3.1.3	General	No pyrotechnics shall be permitted
3.1.6	General	Total stored chemical energy will not exceed 100 Watt-Hours
3.2.10	Mechanical	The maximum mass of a 1U CubeSat shall be 1.33 kg
3.2.10.1	Mechanical	Note: Larger masses may be evaluated on a mission to mission basis
3.2.17	Mechanical	The 1U, 1.5U and 2U CubeSats shall use separation springs to ensure adequate separation
3.3.9.1	Electrical	The CubeSat will have one RF inhibit and RF power output of no greater than 1.5W at the transmitting antenna's RF input
3.4.4	Operational	All deployables such as booms, antennas and solar panels shall wait to deploy a minimum of 30 minutes after the CubeSat's deployment switch(es) are activated from P-POD ejection

Several characteristics are still applicable through the majority of the developed and launched CubeSats projects.

Budget CubeSats are typically missions that are designed and developed allocating budgets lower than those allocated in traditional systems, both for educational projects and for commercial or scientific missions. Standardization, simplicity in the design, reduced and more agile project management and quality assurance efforts, agile approaches to testing, verification and validation, and ultimately limited or no built-in redundancy are causes and consequence of the different approaches.

Launch Traditional satellites are launched into space by dedicated launches. On the other hand, CubeSats exploit their reduced dimensions to secure most of the times launches as secondary payloads, the so-called piggybacking.

Design Thanks to the reduced complexity and standardization of CubeSats projects, less formal design approaches can be employed, and the size and scheduling of the involved teams is often reduced. An increased trend in reducing the documentation packages is also observable.

Modularity One of the key characteristics of the CubeSat ecosystem is the modularity of the technology: several components can be "assembled" to enable functionalities on the platform, resembling a plug-and-play design. This modularity extends to the modularity of the units, where bigger CubeSats can be composed almost by putting together smaller units (Figure 4).

Figure 4 CubeSat modularity is by design one of the key characteristics of the platform. Credits RadiusSpace

COTS A consequence of the trend of reducing costs and extending the reach of the CubeSat standard, is that COTS components have started to populate the majority of educational projects and many of the commercial / scientific ones. Using this type of technology enables low-cost and short implementation cycles, with the added benefit of using latest commercial and industrial grade components. Reduced requirements for reliability of these space systems make the use of non-space-qualified components possible.

Risk CubeSat projects are traditionally characterized by a higher accepted technical risk, that is traded either for a lower cost, a faster implementation, a more favourable approach to innovation, or a combination of these elements. Risk mitigation approaches, even if reduced and more agile, is spreading also in the CubeSat environment.

Market and competition Thanks to the compatibility with non-space-qualified technologies, the CubeSat ecosystem is vibrant with numerous companies providing services and products for the mission designers and developers. This competitive environment is beneficial to the CubeSat technology, as the effects of this competition is the continuous innovation and improvement of the available technology.

2.2.3 The Deployers

As with the evolution of the market and the availability of CubeSat components, the CubeSat deployment technology has seen an increase in the number of available options [4].

Poly-Picosatellite Orbital Deployer (P-POD) It is the original standardised deployer, developed by California Polytechnic State University (Figure 5).

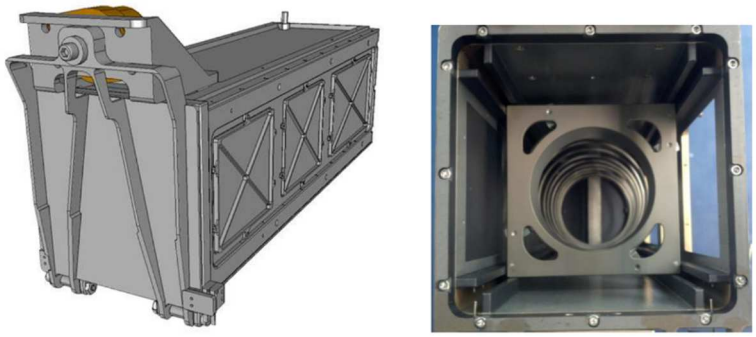


Figure 5 P-POD CubeSat deployer. Credits CalPoly

ISIS Picosatellite Orbital Deployer (ISIPOD) European launcher adapter developed by ISIS – Innovative Solutions In Space.

Japanese Experiment Module Small Satellite Orbital Deployer (J-SSOD) Provides a reliable small satellite launching capability to the International Space Station (ISS). The deployer is handled by the Japanese Experiment Module Remote Manipulator System (JEMRMS), which provides containment and deployment mechanisms for several individual small satellites. The J-SSOD platform is transferred by crew-members into the vacuum of space through the Japanese Experiment Module (JEM) airlock for JEMRMS retrieval, positioning and deployment. The J-SSOD uses a full airlock cycle, with two deployers, to launch a total of 6U.

NanoRacks CubeSat Deployer (NRCSD) It is the first commercial device to deploy CubeSats into orbit from the ISS. It also uses the JEMRMS, but the NRCSD uses two airlock cycles, each one holding eight deployers, each one holding 6U, for a total of 96 Units deployable.

Tyvak Deployers RailPOD Mk.II, NLAS Mk.II, 12U Dispenser, are three deployment solutions developed by Tyvak Inc. Mass optimized and support up to 12U CubeSats.

2.2.4 The Evolution

The CubeSat ecosystem has been object of a distinct evolution in the last two decades, and is interesting to report the status of the technology as of March 2017 (Figure 6). Nanosatellites, despite with some deviation, have maintained the expected forecasts made concerning the adoption of this disruptive technology. Biggest contributions to the increase of the numbers have been private companies and educational projects, as seen in Figure 7.

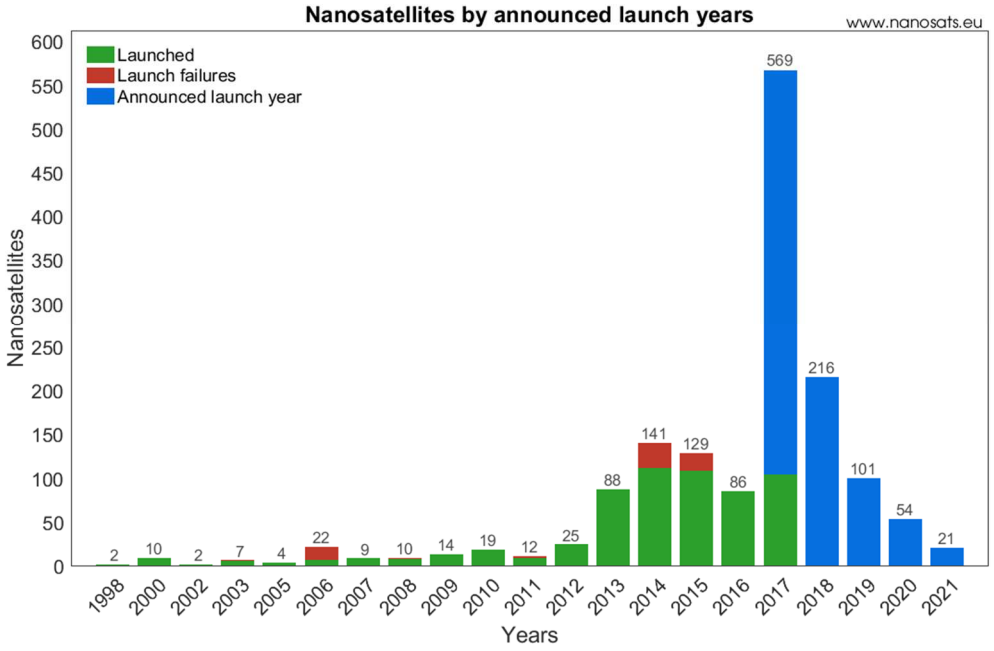


Figure 6 Nano- and Microsatellite launch history and forecast at 2017 (1 - 50 kg). Credits NanoSats.eu

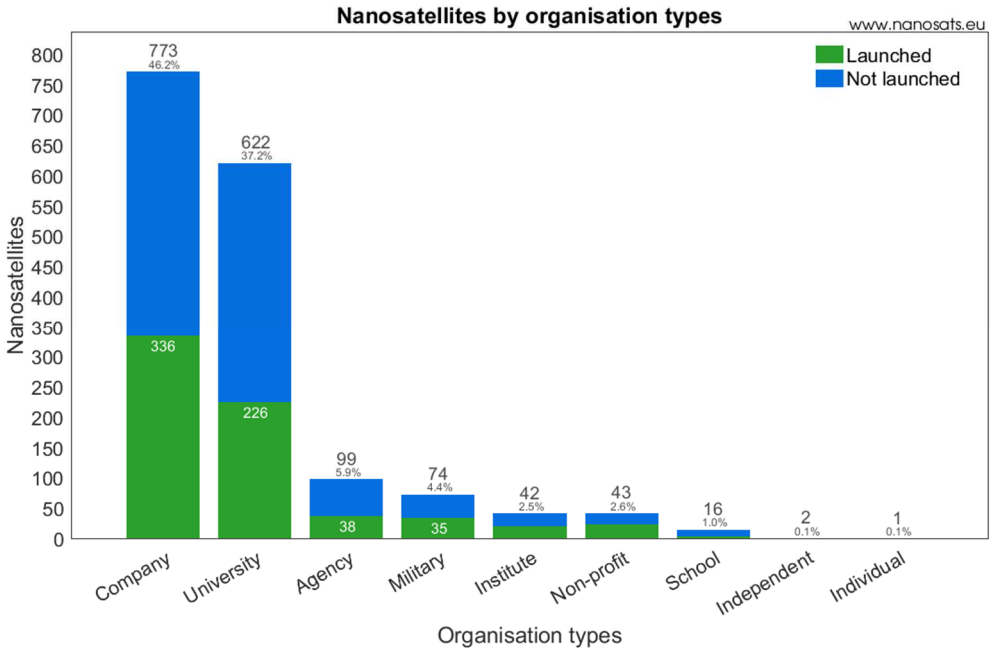


Figure 7 Repartition of the CubeSat projects among organization types. Credits NanoSats.eu

Concerning the diffusion of the CubeSat platform in the world, the repartition sees countries that have already developed a stable space program lead the chart. Despite this, the CubeSat technology has been fundamental in enabling access to space for those countries that did not launch any satellite yet (Figure 8).

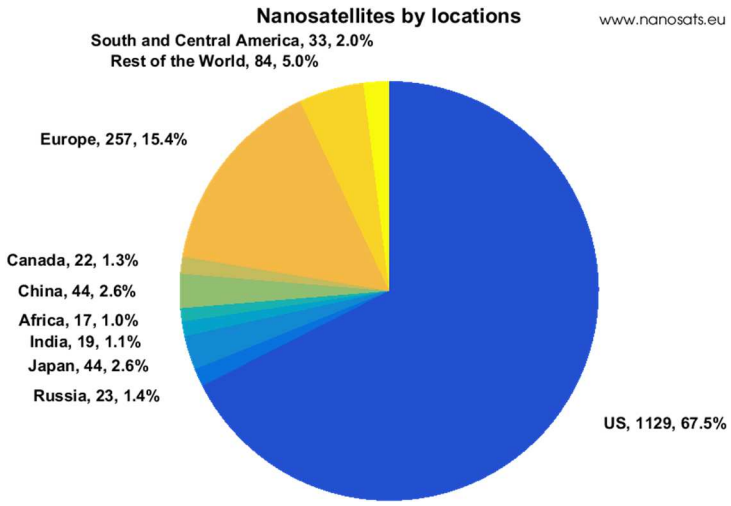


Figure 8 Repartition of the CubeSats per developer nation. Credits NanoSats.eu

As introduced above, one of the key features of CubeSats is the modularity: it is possible to design the space systems in different sizes. Interestingly, the four major sizes (1U, 2U, 3U and 6U) are also the most common choices, with 1U and 3U platforms leading the choice for mission developers (Figure 9). This might be due to concurrent reasons:

- Smaller platforms (1U) often involve lesser costs and more launch availability, therefore enabling more and more entities to develop their own mission
- Increased sizes enable more complex and more performing platforms and payloads. In this sense, 3U and 6U CubeSats are the preferred choice when performances requirements are stringent.

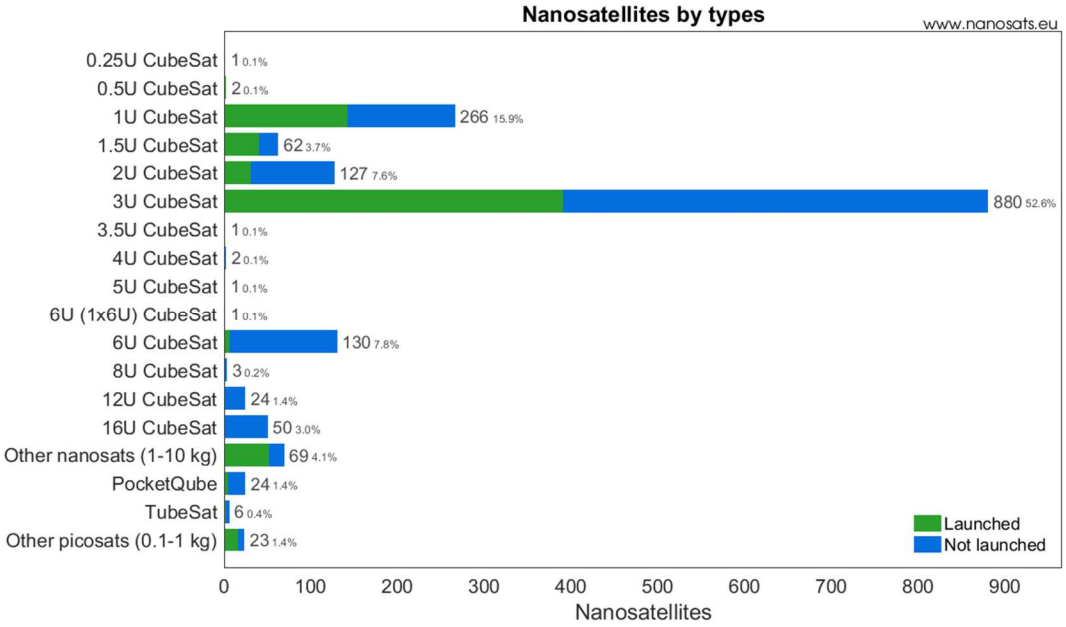


Figure 9 Nanosatellite types are not equally chosen by the mission designers. Credits NanoSats.eu

Despite the high adoption rate, the CubeSat platform is not exempt of problems during the mission: due to the selection of COTS components, to the agile development and testing cycles, the failure rate of CubeSat missions is higher with respect to the traditional ones [18], [19]. Nonetheless, numerous CubeSats have been performing successful operations in orbit (Figure 10).

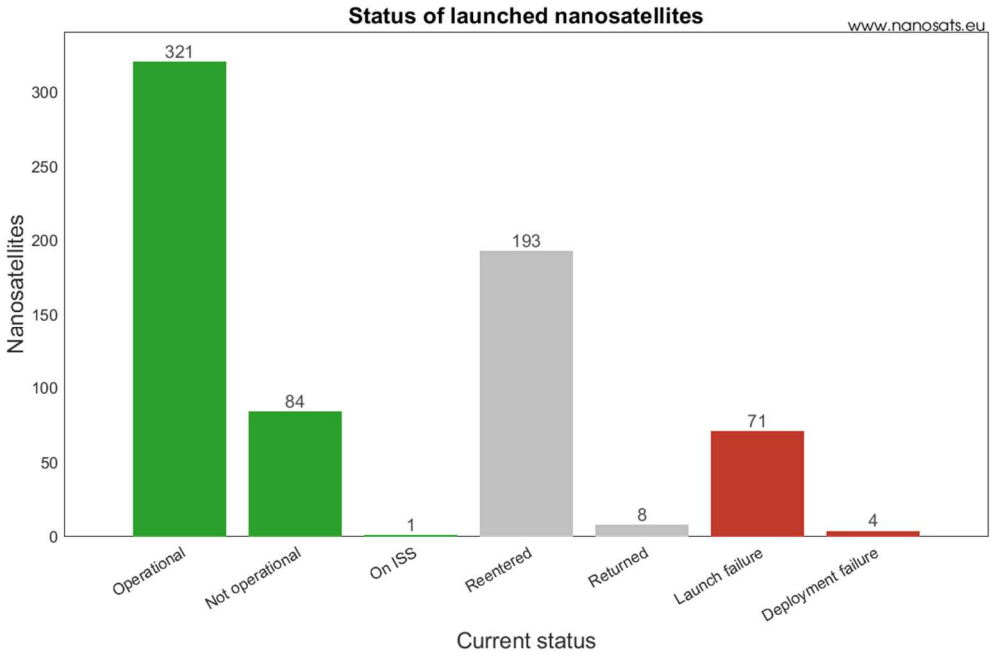


Figure 10 Nanosatellite operational status [16]. Credits NanoSats.eu

2.3 Application scenarios

2.3.1 Historic Small Satellite Missions

Several missions could be cited among the set of historical Small Satellite missions. Here a few important missions are presented.

First CubeSats were launched in 2003 from Plesetsk, Russia, and placed in a sun-synchronous orbit. They were the Danish AAU CubeSat and DTUSat, the Japanese XI-IV and CUTE-1, the Canadian Can X-1 and the US Quakesat. CUTE-1, after at least 9 operational years in orbit, is, among other examples (such as Swiss Cube) one of the longest operating CubeSat mission ever deployed.

SMART-1 was a Swedish-designed, European Space Agency satellite that orbited around the Moon in a mission that lasted 3 years, from the launch in 2003. The acronym stood for Small Mission for Advanced Research in Technology-1. The satellite was used a technology demonstrator for the Hall-effect thruster and other technologies.

PROBA series, are ESA operated satellites designed to host scientific experiments and technological demonstrations. Payloads included hyperspectral

instrument and a black and white camera with a miniaturised telescope. Launched up to 2017 are the PROBA-1, PROBA-2 and PROBA-V.

IPEX is a CubeSat developed and launched by NASA JPL with the objective of validating autonomous operations for onboard instrument processing and product generation. The CubeSat is the first, and probably only, CubeSat implementing state of the art level of autonomy on board. In addition, the CubeSat carried the Continuous Activity Scheduler Planner Execution and Re-planner, to enable mission replanning [20], [21].

2.3.2 Interplanetary CubeSats

The evolution of space systems has progressed without interruption since the Sputnik-I satellite was launched. Improvement in the technologies, in the design and fabrication processes, advancements in the scientific research, innovative mission concepts enabled by successfully reaching previous mission objectives, can be all seen as reasons for the advancement in the performances of the spacecraft platforms and payloads. Some trends are interesting: mission lifetime has, on average, increased through the years (Figure 11); spacecraft bus mass has increased, while payload mass has remained constant (Figure 12). In general, the increased bus mass is connected to higher requirements for mission lifetime, radiation shielding and/or redundancies integrated in the platform. When considering the trends of the various subsystems technologies, the trend is reversed: newer subsystems would perform better and with a lower mass (normalized) with respect to older counterparts [22].

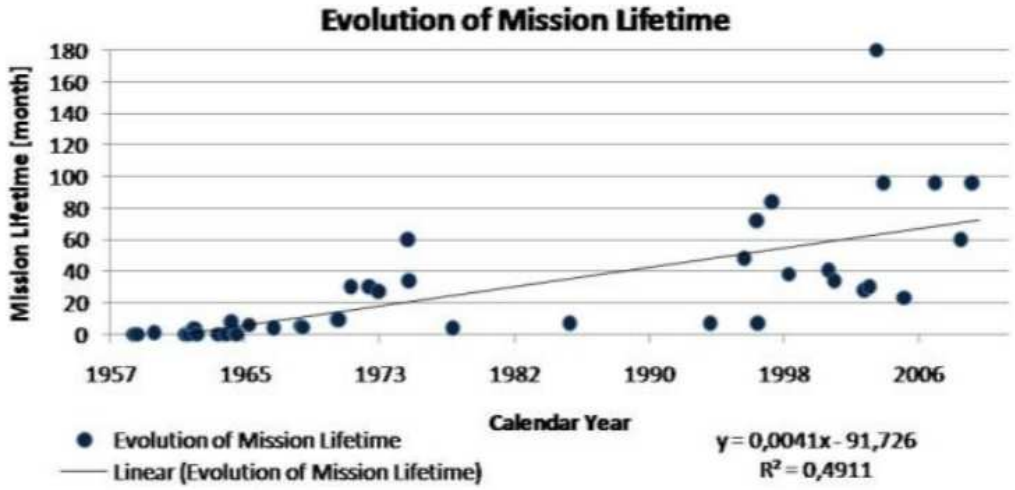


Figure 11 Evolution of mission lifetime. Credits DLR

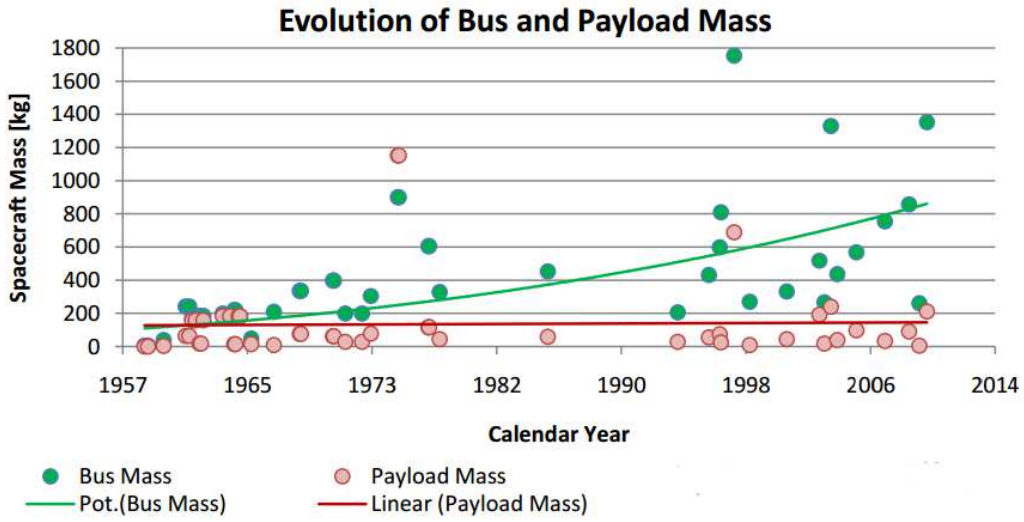


Figure 12 Evolution of Bus and Payload Mass. Credits DLR

This trend is observable also in payload dimensions, where the miniaturization is also playing an important role. Amazingly, as a consequence of the trend, a new category of payloads has started to be considered in mission concept formulation and design: nanosatellites as payloads of flagship, interplanetary missions.

Thanks to the increased capabilities of nanosatellites and to their reduced mass and volume, several innovative mission concepts have begun to appear. The key factor in these mission concept is that a flagship spacecraft would carry one or more CubeSats during an interplanetary mission, to fulfil additional mission objectives and enabling new concepts of operations, by releasing the nanosatellites in situ once the mothership has reached its destination. Example of these concepts are:

AIM mission and its CubeSats to Didymos Binary Asteroid (cancelled) [23]: a spacecraft would release up to 6U total of CubeSats in situ at the Didymos asteroid, to perform technological and/or scientific objectives, either by supporting the main mission or by fulfilling additional goals

CubeSats to Europa: NASA is considering new CubeSat concepts to be deployed to Europa by the Europa Clipper mission [24]. The mission concepts involving CubeSats will have to face interesting design problems: Europa world is considered a potential candidate to host extra-terrestrial life, and therefore contamination will have to be avoided by performing severe sterilization to the CubeSat platforms.

MarCO CubeSats, that will be released by the Insight mission to Mars during the interplanetary transfer from the Earth. The objectives of the CubeSats will be to monitor and record the Entry, Descent and Landing (EDL) telemetry of the Insight probe, and to relay that information back to Earth. Given the high amount of escape velocity, MarCO CubeSats will not be inserted into Martian orbit [25].

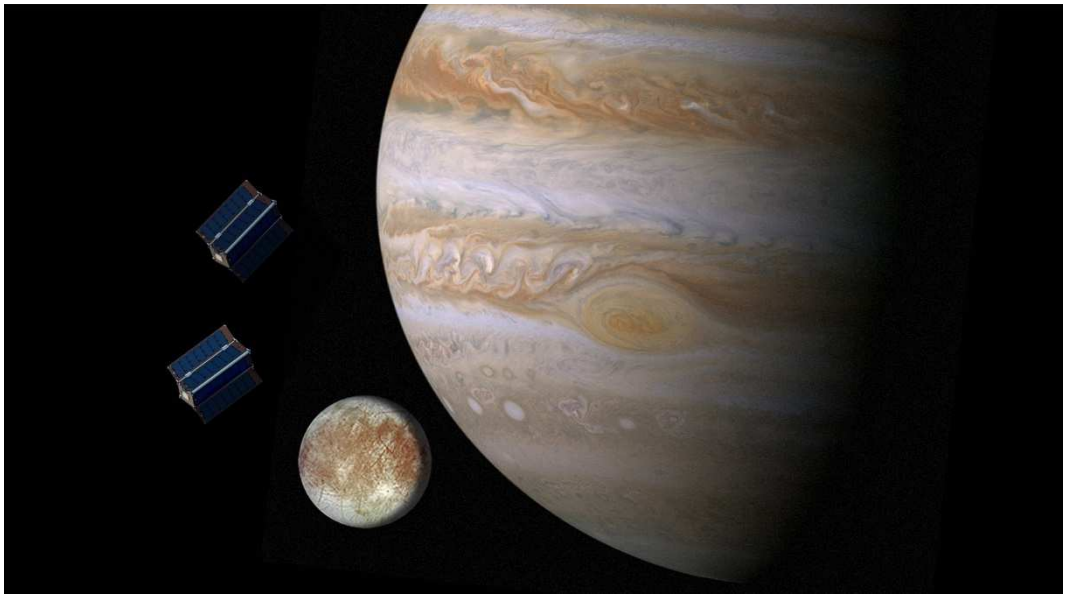


Figure 13 Artist rendering of two 3U CubeSats to Europa. Credits NASA JPL

Interplanetary CubeSats are not only those that are directly released in situ by a mothership. Several concepts have appeared where the CubeSats are released on a transfer orbit by the launcher or by the mothership, and the orbit insertion is performed directly by the CubeSats themselves. Examples of these types of mission concepts are the Exploration Mission 1 (EM-1) secondary CubeSat payloads, that will be released on a Moon transfer orbit by the first mission of the Space Launch System (SLS) [26]:

Bio Sentinel, carrying live organisms in a deep-space mission to assess how they will survive throughout its 18-month mission duration. The Bio Sentinel mission aims at assessing the risks involved with radiation exposure on humans, to prepare radiation protections for future missions.

NEA Scout will perform reconnaissance of an asteroid, taking pictures and observing its position in space. The data collected will enhance the current

understanding of asteroidal environments and will yield key information for future human asteroid explorers [27].

Lunar Flashlight will look for ice deposits and identify locations where resources may be extracted from the lunar surfaces. It will use lasers to reflect sunlight and illuminate permanently shadowed craters at the lunar poles. A spectrometer will then observe the reflected light to measure the surface water ice.

The EM-1 mission (and the SLS in general) will deploy 13 6U CubeSats.

2.3.3 Earth Orbiting Constellations

Another fundamental aspect of the CubeSat ecosystem is that they enable the design and deployment of mission architectures involving a great number of spacecraft for a considerably smaller budget when compared to traditional assets and constellations. In addition, the availability of components enables mass production strategies that are currently not considerable when dealing with bigger systems[1]. Interesting cases of CubeSat constellations are here presented that are currently disrupting the spacecraft and the space data market.

PlanetLabs is a constellation of CubeSat to be deployed to LEO, designed for Earth Observation (EO). It is constituted of several 3U CubeSats that are usually deployed on piggyback launches [28]. The company exploits the great scalability of the CubeSat technology to perform unprecedented EO, with over a hundred satellites in operations. In 2017, the company performed a record-breaking launch of 88 satellites [29].

Planetary Resources is an American company focused on advancing humanity technology level to enable asteroid mining, to exploit the incredible amount of resources that are available in these celestial bodies. Initially aiming at performing asteroid mining operations, the company has recently secured launches for the Arkyd-100 series of space telescopes [30].

Spire Global is an American company whose aim is to deploy a CubeSat constellation, initially thought to be made of 125 satellites, that host a GPS radio occultation payload and a AIS signal tracking payload [31], [32].

[1] Possibly the sole case of medium-sized satellite constellation up to 2017 is OneWeb constellation.

OneWeb is one of the most ambitious constellation projects that are currently under development. It features a total of 648 operational satellites in 18 orbits at 1200 kilometres of altitude. Each small satellite will weigh between 175 and 200 kg in mass. The mission objectives are to provide internet broadband connectivity with a worldwide coverage [33].

2.3.4 Other relevant cases

These concepts are not always adhering to the Small Satellite or CubeSat standards, but are nonetheless interesting as they share a similar philosophy: reducing sizes to enable new mission architectures.

Mars Helicopter, a concept developed by NASA JPL, highly resembles the CubeSat form factor. The helicopter would be used to pinpoint interesting targets on the Martian surface, effectively tripling the rover driving speed [34].

Copernicus Master Small Sat is a competition introduced in 2017 by the AZO organization, for the design, development and launch of a Small Satellite to support Sentinel satellite missions.

KickSat 1 was an innovative CubeSat mission released for crowdfunding on Kickstarter in 2011, with the aim of releasing hundreds of sprites (small chipsats equipped with a radio, solar cells, and microprocessor), that would beacon customized messages defined by the crowdfunders [35]. The satellite failed to deploy the sprites.

Mars and Lunar Penetrators are concepts of ground-penetrating systems intended to be released as impactors on a re-entry trajectory towards a celestial body, with the aim of penetrating the surface and to study the underlying substrate. Concepts were formulated both for the Moon and for Mars missions [36].

Chapter 3

Space Mission Software

3.1 Overview of Flight Software

A spacecraft Flight Software (FSW) is generally designed to perform very specific (and often mission-unique) functions, and, on the other hand, it has to satisfy very diverse, and often competing, needs. Throughout the years, the FSW has become the traditional interface between the GS and the spacecraft, and with the arrival of new technologies, new functionalities of the FSW have also to be implemented.

In general, a typical FSW will have the following functionalities.

Command & Data Handling related functions:

- Executive and task management
- Time management
- Command processing
- Engineering and science data storage and handling
- Data monitoring
- Fail safe and safe mode
- Failure Detection, Isolation and Recovery

Attitude and Orbit Control related functions:

- Attitude determination and control
- Orbit determination and navigation

- Orbit management
- Propulsion

Other bus related functions:

- Communication management
- Electrical power management
- Thermal management

Payload related functions:

- Payload data commanding
- Payload data management
- Payload calibration

It has to be noted that the different algorithms constituting the FSW might be physically located in different subsystems: the AOCS software might run in the AOCS board, the COMSYS software on the COMSYS board and so on, depending on the location of the different microprocessors or microcontrollers.

3.1.1 Command and Data Handling

The C&DH for typical small satellite missions, especially for CubeSat, traditionally features standardized characteristics. Among these, an Operating System (OS), that has the objective of handling the low-level interfaces with typical components of a processing board: storage, RAM and ROM, interrupts, and so on. Typical OS for Small Satellites and CubeSats are: Linux, RTEMS, VxWorks, FreeRTOS, Salvo [37]. The OS software is generally divided into layers (Figure 14). In order to streamline the process of development of Small Satellite projects, the mission developers are moving towards coding applications in the higher layers of the architecture, leaving lower level coding to the Original Equipment Manufacturer (OEM). The C&DH middle to higher layers include decision-making algorithms, time management, command processing, engineering and science data storage, and higher-level communication functions. In general, C&DH is the coordinating core of all the on-board processing, apart from some localized data management.

Data is managed and stored on specific memories, that in the C&DH for Small Satellites assume the form of SD and microSD cards, that are now reaching very promising levels of performance, storage capacity and reliability: extended temperature ranges, radiation and magnetic field resistance.

Typical programming languages for the highest level of a CubeSat FS are:

- C, C++: traditionally one of the most used programming languages for embedded applications, thanks to the extreme control available to ensure efficiency and to predict performances, while still maintaining good readability
- Python: especially suited for CubeSat applications, features rich library availability, and is one of the most immediate languages to learn. This is beneficial especially for low-budget, educational projects, where training of the personnel (or students) must be performed as quickly as possible. Drawback of using Python, as with other garbage-collected languages such as Java, is the presence of fairly unpredictable latency peaks, that make the use of these languages less indicated for Hard RT applications.

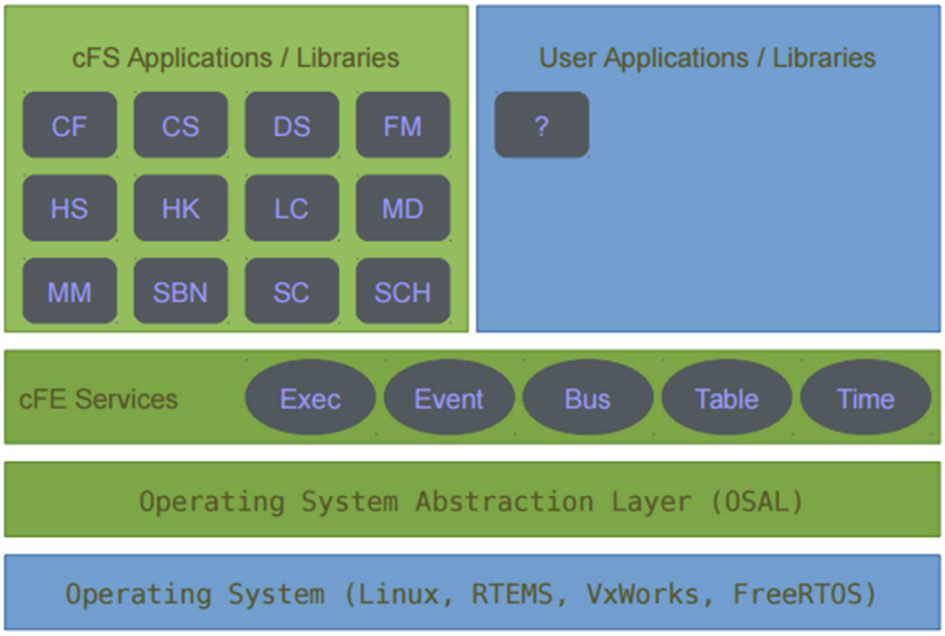


Figure 14 Example of Operating System layers: core Flight Software. Credits NASA

3.1.2 Other software

Payload and Instrument processing software

Often distributed in several processors and controllers in the spacecraft, this type of software has usually very specific applications and is rarely reused among different

spacecraft with different payloads. Autonomy enhancing algorithms have interesting applications in this category, both from a data reduction perspective and from an event detection one.

Failure Detection algorithms

The type of algorithms involved with monitoring of failures is traditionally highly distributed, with detection of errors performed locally, with increasing centralization of the computation the higher the level of abstraction of the reasoning. Fault correction is typically centralized and abstracted, in order to deal with different and multiple types of failures in similar ways.

Microprocessors and other computing units

On a typical spacecraft, several computing units might be present: systems-on-chip, microcontrollers and microprocessors might be spread in several boards or subsystem of a spacecraft, effectively achieving a distributed architecture. In general, few units assume a leading role in managing the whole spacecraft: common architectures employ as much as three units for general management, scheduling and so on. Other chips usually perform very specialized tasks and their reach on other subsystems is very limited. These types of computing unit usually employ low-level programming languages, such as Assembly. Examples of such applications can be microprocessors to manage the peripherals of a telecommunication board, microcontrollers to pre-process instrument data, and so on.

3.2 Overview of the Ground Software

Since the beginning of the space era, several iconic tasks and actions have been performed by the Mission Control centre, before the spacecraft started to be capable enough to substitute it: planning and scheduling, communication link establishment, science data management, calibration, Health and Safety verification.

These functions were so important that the MC quickly became involved with the highest responsibilities for managing the spacecraft and its activities, yet relying on the space segment to provide most of the information needed to perform the MC duties. In general, when designing spacecraft operations, it is important to consider all the functions that will have to be performed, irrespective of the location (GS or SS) that will execute them.

3.2.1 Planning and Scheduling

Planning and Scheduling (P&S) is one of the most important tasks when operating a space mission: the generation of detailed desired optimized timeline of spacecraft activities. These activities can sometimes be based on complex modelling of the spacecraft environment and of the expected behaviour: examples of these are the Hubble Space Telescope and the Kepler mission. Once the schedule is defined, it is uploaded to the spacecraft and executed in a time-tagged way. In general, the definition of the activities is performed not only for the nominal path, but alternate branches of off-nominal conditions are also foreseen and generated. Interestingly, the definition of the timeline of operations is a process as time-dependent as the execution of the operations itself: in certain cases, the look-ahead period can reach several months to one year. Historically, long term operations definition is performed to constrain the choice of medium-term to immediate operation definition in given periods of the mission (Sun-Earth-Spacecraft geometry is an important factor [23]). On the medium term, events such as the South Atlantic Anomaly entry/exit or similar events are accounted for. On the short term, final detailed scheduling to precision of seconds is defined using the most accurate available data. The process of operation definition has traditionally been very iterative. A considerable progress has been made with the intent of making this process more flexible and efficient, yet some inefficiencies and complex modelling are unavoidable.

3.2.2 Command Loading

Command loading is one of the fundamental functions that most, if not every, spacecraft mission has performed at least once. In general, this activity has become straightforward. It consists in converting the P&S outputs into specific commands understandable by the spacecraft FSW. Automation is increasing in this domain.

3.2.3 Science Scheduling and Support

Science activities execution is traditionally constrained on the spacecraft, while a consistent amount of work is required to plan the scheduling and for the support activities: highly specific mission and science instrument activities that may include calibration, management and direction of operations. Calculations to support the definition of these activities can take a big amount of human resources.

3.2.4 Failure Detection

When a spacecraft encounters issues on board, the systems engineers on ground must perform a diagnosis using the telemetry downlinked to ground. In these cases, operations personnel must either rely on their skills and experiences, or use tools to support the failure mitigation task. One of the main results of employing advanced tools is that they vastly improve the speed at which the failure identification is performed. Artificial Intelligence, performing pattern recognition, is one of the best candidate for this type of task.

3.2.5 Data Analysis, Calibration, and Processing

In general, nearly all spacecraft engineering analysis and calibration functions have been performed on ground. These include attitude-sensors alignment and polynomial calibrations, battery depth of discharge and state-of-charge analysis, communications margins evaluations and so on. There does not seem to be a clear cost difference if these functions are performed on ground or on board. In addition, science data processing and calibration have been nearly exclusively a ground system responsibility for two main reasons: limited on-board computational capabilities of rad-hard processors and a bias in the scientific community that insisted on having all the scientific data downlinked to ground. There is still a strong opinion that science data might not be processed as thoroughly on board as it is on ground, and that science data users often process the same data multiple times using different algorithms, calibrations and so on, even years later after the data were downlinked.

It is still advisable to design missions with autonomy levels that do not force the science users to rely on decision taken only on-board, but rather offer the option to receive processed data or instead the complete set of acquired data.

3.3 Flight vs Ground Design

Autonomy level are steadily increasing thanks to improved and more reliable flight system hardware capabilities (computational power, hardware input/output handling, storage capacity, and so on), and to innovative approaches to the design of the FSW architecture (object-oriented design, expert systems, remote agents and so on). Moreover, specific approaches and operations that were intended explicitly for the Ground Segment are now moving towards the Space Segment: engineering data analysis and calibration, science processing and calibration. The result is that

spacecraft have more and more the capability of taking advantage of the strengths inherent of a RT software system in direct contact with the flight hardware.

The most striking characteristics of the FSW with respect to its ground operation counterpart are:

- Reaction times
- Completeness
- No delayed information

Only the FSW of a spacecraft directly located in situ at the mission can instantly access flight HW measurements, process the information and act in real-time.

An example might be obtained considering the AOCS: only the on-board computer has complete and immediate access to the spacecraft status in realtime, obtaining critical information well before a ground-based operator could.

On the other hand, previous approaches have assigned more importance to the Ground Segment of a space mission, thanks to more powerful ground computers that have allowed the Mission Control to execute complex schedule optimization algorithms using highly complex predictive models. Even if the computational power of both the ground-based systems and the spacecraft ones is increasing, and somewhat narrowing, improvement potentialities exist also on the Ground Segment [38], [39].

Chapter 4

Mission Autonomy

4.1 The problem of Autonomy

Since the beginning of the space age, a trend has become evident: with the improvement of the experience and the technology associated with a mission, came the desire and the need of more sophisticated mission. New instruments and payloads are being developed, with increasing capabilities of data collection. In addition, new worlds, new science, and new phenomena to observe are appearing on the horizon. The new scientific goals and objectives often require multiple coordinating spacecraft to make simultaneous observations, or to detect events without ground intervention. This increase in the demands for new spacecraft has led to intense research and development efforts for the software applications and processes that are used during a space mission, both on ground, in the Mission Control centre, and on-board, integrated into the Flight Software.

One of the key drivers for enhancing the capabilities of spacecraft for remote and complex missions has also been the fact that human exploration missions have received a setback, due to increased security standards in the design directions [40]. It is currently not advised to consider human exploration in certain kinds of mission, for example mission to asteroids. In addition, several issues impede the deployment of astronauts even in less exotic mission concepts: long mission timelines due to the distances involved, or the radiation environment. More and more, there is an evident necessity to develop unmanned missions with respect to manned ones.

The present chapter present and discusses Mission Autonomy and its management. On-board autonomy management addresses all the aspects of the functions performed by the spacecraft that give the capability to fulfil mission objectives (by performing certain operations) and to survive critical situations without relying on ground segment intervention.

4.2 Key concepts: Automation, Autonomy, Autonomicity

Before proceeding, it is important to understand the differences between automation, autonomy and autonomicity, as these concepts are used in space engineering, but they have very different applications and characteristics. These concepts refer to actions executed without any human intervention from the beginning to the end. *Automated* processes follow, in a step-by-step fashion, a routine that replaces manual processes and that might still involve human cooperation. *Autonomy*, on the other hand, involves operations that have the goal of emulating human thought processes, rather than just substituting them [41]. *Autonomic* processes, at last, involve processes in the area of self-awareness and self-management.

An example of automatic process, related to spacecraft operations, would be a spacecraft that turns on a payload and performs initial checks, in a series of operation steps. In general, on-board procedures could be assimilated into the *automatic operations* label. Another example would be a process that regularly extracts from the data storage a set of telemetry parameters, performs a standard statistical analysis of the data, outputs in report form the results of the analysis and generates appropriate alerts of identified anomalies. Moreover, an automatic process performs no independent decision-making based on real-time events, and a human operator is required to respond to the outcome of the routine [42].

An example of autonomous process on ground would be a program that monitors the spacecraft position in the orbit, determines when the communication is possible, determines which files to uplink and sends them, accepts downlinked data, verifies them and request retransmission if necessary. A flight software example would be a software that, by processing the data obtained by a IR camera, senses that there is a forest fire in the area observed by the satellite, and decides to allocate more observation time to that particular area, instead of continuing the observation plan [21].

Key characteristics of autonomic traits are linkable to reflexes found in nature, and to spontaneous behaviours. In particular, four properties related to self-management are assimilable to autonomic computing:

- Self-configuring
- Self-healing
- Self-optimizing
- Self-protecting

These four traits are often associated to four properties:

- Self-aware: internal capabilities and state of the managed components or equipment are known to the system
- Self-situated: the system has awareness of the external environment and context
- Self-monitor and self-adjust: through monitoring sensors, actuators and control loops

Table 3 How the three levels are defined among different entities

Intelligent Machine Design	**Future Communication Paradigms**	**DARPA/ISO's autonomic information assurance**	**NASA's science mission**	**Self-directing and self-managing system potential**
Reflection	Knowledge plane	Mission plane	Science	Autonomous
Routine	Management control plane	Cyber plane	Mission	Self-aware
Reaction	Data plane	Hardware plane	Command sequence	Autonomic

Machines infused with Artificial Intelligence, to autonomously operate in their specified environment, are traditionally constituted by three layers of behaviours: a top level one, linked to reflection; a middle level, enabling reasoning routines; and a bottom one, enabling reactions. At the reaction level, no learning occurs, but

immediate responses are performed as a reaction to state information coming from sensors. The routine level is where evaluation and planning are performed. Receives inputs from sensors and both from the reaction and the reflection level. At last, the reflection level receives no sensor input and has no output to actuators: receives inputs from the reasoning level and the reaction level, and performs reasoning about the state of the machine itself.

4.3 Autonomy versus Costs of Missions

Another direct effect of implementing more sophisticated mission operations management software (either on-board or on ground) can be highlighted analysing the costs of the mission, both in the total amount directly impacting the budget, and on the repartition of the costs in the various activities. All the main space agencies have allocated significant efforts in reducing the human-supervised operations in favour of automating spacecraft functions. The current approach, both for designs and methodologies, involves spacecraft downlinking their mission data (both health keeping and payload) to Mission Control for processing, and Mission Control centres uplinking commands to the spacecraft. As the complexity and number of spacecraft increase, it takes a proportionately large number of personnel to control the spacecraft [42].

Table 4 Example of spacecraft constellation and the relative human resources needed for control. WMAP: Wilkinson Microwave Anisotropy Probe, NMP: New Millennium Program; MC: Magnetotail Constellation

Mission	Year	Number of spacecraft	Operators needed with current technology	Current people per S/C	Goal people per S/C
WMAP	2000	1	4	4	-
Iridium	2000	66	200	3	-
GlobalStar	2000	48	100	2	-
NMP ST5	2007	3	12	-	1
MC	2012	30-40	120-160	-	0.1

Table 4 illustrates some constellations (proposed or flown) and compares the amount of HR needed to operate the mission with present technology and future technology [43]. Missions capable of fulfilling the desired science objectives will obtain the operator-to-spacecraft ratio objectives only if designed to operate without constant control and commanding by MC. The amount of HR considered in the last column of the table will require substantial development effort in the autonomy segment of the mission. In general, it is expected that, for multi-spacecraft missions, featuring tens or hundreds of satellites, operations will be impossible to be carried out without near-total mission autonomy.

4.4 History of Autonomy Features

4.4.1 Up to 1980

This period saw the first efforts into standardizing FSW, and the appearance of the first automatic actions performed by a spacecraft. In particular, earliest efforts in automating operations came on the HEAO series of spacecraft, with some automatic functions such as pointing control, limited failure detection, stored commanding and telemetry generation. Additional commanding capabilities included the now standard absolute-timed, relative-timed and conditional commands. Limit checking as FDIR was also implemented, with automatic mode transition to pre-programmed safe modes. On the Solar Maximum Mission (SMM), an embryo of autonomous target identification and acquisition capability was implemented, that would be later refined into Hubble Space Telescope (HST). SMM processing algorithms could detect solar flares, and re-program spacecraft pointing to observe the phenomenon. This characteristic was also present in the Orbiting Solar Observatory-8, launched in 1975: it could steer its payload platform independently to perform observation of its targets.

The evolution of on-board pointing capabilities can be seen just by looking at the pointing independence of the two spacecraft, HEAO-1 and HEAO-2: the first one relied on attitude reference updates every twelve hours based on ground attitude determination. The follow-on spacecraft, two years later, already possessed the capability to compute its own attitude reference update, based on ground-supplied guide-star reference information, a capability also implemented in SMM. HEAO-2 could, in addition, periodically go through a weekly target list.

4.4.2 1980-1990 Spacecraft

The 1980 saw the launch of larger, more expensive and more sophisticated spacecraft. Among these, some famous spacecraft such as the HST and Compton Gamma Ray Observatory (CGRO) were actually launched in the 1990s, but were scheduled to be launched earlier.

HST featured automatic safe mode options and improved FDIR checks; and the first appearance of “message based” architecture between two processors, that would coordinate when searching a new observation target. Moreover, it has to be noted that many of the advanced FDIR functions of the HST were added to the spacecraft after launch, in response to problems experienced inflight.

Figure 15 Hubble Space Telescope. Credits NASA

Another exemplar mission was the Extreme Ultraviolet Explorer (EUVE), that featured innovative telemetry monitoring capability and autonomous generation of commands. In addition, the spacecraft was integrated with a predecessor of a true event-driven operation reasoning engine.

4.4.3 1990-2000

The spacecraft developed in this decade were characterized by HW and SW enhancements: on-board computers were more powerful, more RAM and more storage was available on-board, and on the software side new higher-level languages (as C, C++ and Ada) and floating-point arithmetic allowed the FSW to assume characteristics comparable to those of ground software.

Autonomy advancements featured better interconnection between different processing units and different SI in the spacecraft. Moreover, the decoupling of the science and communications scheduling introduced further flexibility in spacecraft. Additional features concerned the telemetry definition tasks, that are now configurable directly by table uplink, and this allows to reprogram the spacecraft telemetry without changing the FSW. Advanced decision-making was also implemented thanks to the introduction of Boolean logics to correctly isolate failures (Landsat-7). In these years, spacecraft such as Deep Space One (DS-1) were launched and later integrated with Remote Agents, responsible for multitasking, P&S and model-based FDIR [44].

4.4.4 2000s

Among the new capabilities implemented on spacecraft in the 2000s, true lost-in-space capabilities can be highlighted, along with even more improved model-based failure detection. In general, the trend observed is moving towards the implementation of SI acting as spacecraft controllers themselves, deciding autonomously the science schedule with respect to planned and unplanned observations.

Additional experiments in autonomous formation flying have been performed.

4.4.5 Current and Future Spacecraft

Spacecraft under development (such as the James Webb Space Telescope), are implementing advanced features such as on-board event-driven scheduling, with a flexible implementation that allows to move through observation targets as soon as they are available, without forcing any observation if anomalies or unfavourable conditions appear.

Developments in spacecraft constellation and formation flying are currently driving the effort in mission autonomy research. Another important driver is the

independence of SIs with respect to the spacecraft pointing. Finally, innovative, AI-driven small spacecraft are being flown [20], [45].

4.5 ESA Autonomy Design Guidelines

The design and implementation of autonomy features on-board is yet to become standardized. On the other hand, guidelines and requirements that cover the autonomous operability of a spacecraft have been already laid by ESA, and are available to the spacecraft manufacturers [46].

In general, the design of the on-board autonomy should take into account high-level operations characteristics such as:

- Maximum level of mission outage that is considered acceptable
- Ground Control Station access durations and timings
- Maximum period of ground segment outage to be foreseen

Certain values, characteristics of each mission, should be defined when designing a space mission:

- An autonomy duration, that is the time the spacecraft can continue operations without instructions from ground
- A storage duration, that is the maximum time interval that the spacecraft can continue storing new mission data, without downlink and subsequent erase
- A maximum time during which the spacecraft can autonomously manage its operations in the presence of a single failure. It also includes the time spent by the Mission Control to detect, identify and plan the recovery action for the failure
- The design of the spacecraft behaviour in the presence of a failure shall take into account a minimum reaction time of the Mission Control
- The Mission Control, through defined Ground Control Stations, should be able to override any on-board autonomous function.

When designing the autonomy features of a spacecraft, several application scenarios must be considered: nominal operations, off-nominal operations and data management autonomy.

4.5.1 Nominal mission operations autonomy levels

During the execution of nominal mission operations, four levels of autonomy have been defined:

- Execution mainly under real-time ground control
- Execution of pre-planned mission operations on-board
- Execution of adaptive mission operations on-board
- Execution of goal-oriented mission operations on-board

These autonomy level, and their features, are summarized in the following table.

Table 5 Mission execution autonomy levels

Level	Description	Functions
E1	Mission execution under ground control; limited on-board capability for safety issues	Real-time control from ground for nominal operations Execution of time-tagged commands for safety issues
E2	Execution of pre-planned, ground-defined, mission operations on-board	Capability to store time-based commands in an on-board scheduler
E3	Execution of adaptive mission operations on-board	Event-based autonomous operations Execution of on-board operations control procedures
E4	Execution of goal-oriented mission operations on-board	Goal-oriented mission re-planning

4.5.2 Mission data management autonomy

Concerning mission data management, the following autonomy levels have been defined:

- Essential mission data used for operational purposes can be stored on-board
- All mission data can be stored on-board (science data and housekeeping data)

The following table summarizes the details of these autonomy features.

Table 6 Mission data management autonomy levels

Level	Description	Functions
D1	Storage on-board of essential mission data following a ground outage or a failure situation	Storage and retrieval of event reports Storage management
D2	Storage on-board of all mission data, i.e. the space segment is independent from the availability of the ground segment	As D1 plus storage and retrieval of all mission data

4.5.3 Fault management mission autonomy

Failures are a fundamental aspect of each space mission, and the correct management of expected and unexpected failures is often the line between a successful mission and an unsuccessful one. Generally speaking, the approach towards the management of failures is the Failure Detection, Isolation and Recovery (FDIR) approach. In this scope, failures are managed in the following way:

- They are detected (on-board or on ground) and are reported to the relevant subsystems/systems and to the Mission Control
- They are isolated, that is the propagation of the failure among other components/subsystems/systems is inhibited
- The functions affected by the failure are recovered, to allow for mission continuation

The following autonomy levels have been defined:

- Autonomy to safeguard the space segment or its sub-functions
- Autonomy to continue mission operations

These levels are described more into details in the following table.

Table 7 Failure management autonomy levels

Level	Description	Functions
F1	Establish safe space segment configuration following an on-board failure	Identify anomalies and report to ground segment Reconfigure on-board systems to isolate failed equipment or functions Place space segment in a safe state
F2	Re-establish nominal mission operations following an on-board failure	As F1, plus reconfigure to a nominal operational configuration Resume execution of nominal operations Resume generation of mission products

4.6 The need of Autonomy

The potentialities of Small Satellites are clear, and several innovative mission architectures could be enabled by the diffusion and adoption of this category of spacecraft. Unfortunately, as introduced earlier, there are several mission-level and system-level issues that impede the capabilities of small spacecraft especially when applied to complex mission architectures, both interplanetary and Earth-based. The main issues are presented in the following sections.

4.6.1 Multi-spacecraft missions with respect to Monolithic missions

For certain types of scientific or technological goals and objectives, implementing a constellation with respect to a monolithic architecture can bring several advantages:

- Risk spreading among several assets, preserving the chances of fulfilling the mission in case an instrument or system fails
- Performing multiple observations, either in controlled formation flying or in an uncontrolled swarm, of a mission target at the same time from multiple locations
- Distributing different payloads among different spacecraft allows to reduce the complexity and size of each asset
- Replacing an instrument by launching a new spacecraft into an existing constellation or swarm

Missions are currently being planned and proposed that consider tens and hundreds of assets in the space segment. In order to avoid excessive cost of operations, the most promising way is to reduce the operators-to-spacecraft ratio. An important conclusion can be drawn from the last statement: mission operations design, and the operators themselves, need to work at a higher level of abstraction and be able to monitor and control multiple spacecraft simultaneously.

Another benefit of increasing the level of autonomy on a spacecraft is that several subsystem sizes can be reduced, as the performances needed to fulfil the mission might be reached by a synergy of several spacecraft, instead of allocating all the performance on a single one. Among the subsystems that are affected by the autonomy of the space segment is the communication system: introducing higher autonomy features enables the reduction of the downlinked data. Command and Data Handling (C&DH) is another affected subsystem: the increase of the acquired data would require additional on-board storage. This requirement can be mitigated by enhancing the autonomy level, and implementing algorithms that analyse, choose and discard non-meaningful scientific and mission data. On the other hand, the C&DH will be affected by enhancing the on-board autonomy by a likely increase in the computational power requirements of the subsystem.

4.6.2 Big Distances, Low Data Rates and Communications Delays

Another key reason to implement advanced mission autonomy software is the fact that, for certain types of missions, the communication between the MC and the spacecraft takes minutes, if not hours. In these architectures, the mission risks increase because the monitoring of the spacecraft cannot be performed in real-time (or near real-time).

On the same side, another issue impedes the correct fulfilment of space missions: for those mission whose objectives are to study randomly appearing events (for example a comet plume, or a forest fire), the decision time for a human operator is often too long to update correct observation commands to the spacecraft. In this case, the communications delays might be small, but decision-making delays are added, and the result is still a poorly performing mission. Autonomy can play an important role in these cases, because it enables real-time decision-making and a corresponding action can be taken to observe the desired phenomenon. Challenges in this application include the definition of rules to manage the observation schedule, to understand whether it's more important to interrupt current objective (to perform the observation of the newly appeared event) or to ignore the event and continue with the objective in place. An example of this feature is the Swift mission, for which one of the instruments has software functions that determine whether a new observation has high priority, and if so, commanding of the spacecraft can be executed to continue the observation.

At last, large communications latencies are also problematic for failure management: long delays would introduce high uncertainties about the current status of a spacecraft, putting at risk the success of the mission, but also complicating the response of human operators, that would have to take decisions without knowing into details the situation.

4.6.3 Variable Ground Support

Traditional mission design involves carefully planned Ground Segment resource allocation, that allows the mission to be controlled and managed smoothly. This is not always the case: one key example being the category of university CubeSats, especially educational, low-budget projects. In this case, often times the required Ground Control Centre is not available, has poor performances, or there are not enough operators to guarantee a high percentage of presence during satellite passes. A high level of autonomy on the spacecraft would allow the mission to continue

without interruption for long periods of time, determining on its own the best strategies to acquire new data, and to downlink the stored data once a passage is available.

Additionally, there might be missions where complete autonomy may not be the best solution, or that different periods may require different levels of autonomy. In this scenario, adjustable autonomy can be implemented. The adjustment can be performed autonomously by the system, depending on the conditions, or on request by the MC to help the spacecraft accomplish current objectives, or to override the on-board intelligence to perform manual commanding. With adjustable autonomy, it is mandatory to have a well-designed Ground Segment and a robust operation management to work flawlessly with the on-board software.

Chapter 5

Artificial Intelligence

5.1 What is Artificial Intelligence

Artificial Intelligence is a branch of Computer Science that has gained enormous popularity in the last decade, thanks to the many successful applications developed. The term was coined just after the second World War II, in 1956 [47]. Currently, the field is composed by a great variety of subfields, ranging from learning and sensing the stimuli, to specific activities, such as playing games, proving mathematical theorems, writing or even driving and diagnosing diseases. Artificial Intelligence is a universal field, as universal is the range of human activities.

5.1.1 Definitions of Artificial Intelligence

The definitions of this field of computer science are numerous, due to the fact that the field has evolved quickly through the years, and defining with a univocal set of words a field this vast is certainly open to opinions and different point of view. In the literature, eight typical definitions are accepted, each one carrying slightly different meaning and emphasizing certain aspects of the field. An interesting table is provided in [47], and is presented here entirely:

Table 8: Definitions of Artificial Intelligence

Thinking Humanly	Thinking Rationally
"The exciting new effort to make computers think …*machines with minds*, in the full and literal sense." (Haugeland, 1985)	"The study of mental faculties through the use of computational models" (Charniak and McDermott, 1985)
"[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning …" (Bellman, 1978)	The study of the computations that make it possible to perceive, reason, and act." (Winston, 1992)
Acting Humanly	**Acting Rationally**
"The art of creating machines that perform functions that require intelligence when performed by people" (Kurzweil, 1990)	"Computational Intelligence is the study of the design of intelligent agents." (Poole, 1998)
"The study of how to make computers do things at which, at the moment, people are better." (Rich and Knight, 1991)	"AI… is concerned with intelligent behaviour in artefacts." (Nilsson, 1998)

5.1.2 The various philosophies of Artificial Intelligence

These definitions highlight four approaches for implementing an intelligent system.

Acting humanly

Traditionally, designing machines to emulate human way of acting imply giving the machine at least one of the following characteristics:

- Natural language processing – enabling successful communications
- Knowledge representation – storing knowledge
- Automated reasoning – answering questions and drawing new conclusions
- Machine learning – adapting to new situations

- Computer vision – perceiving objects
- Robotics – manipulating objects and moving

Several competitions and tests are held every year in which machines compete in disciplines involving one or more of the listed categories. Moreover, a rigorous and widely famous test on machine capabilities of emulating humans is the Turing test, designed by A. M. Turing.

Thinking humanly

This is an approach to develop Artificial Intelligence focused on defining and implementing the way humans think: the cognitive modelling. This approach is driven by an interaction between computer models from AI and experimental techniques from psychology. Additional effort is put also on emulating the reasoning steps, not only reaching a predefined reasoning output from certain conditions. In general, cognitive science is based on experimental studies performed on real humans or living beings.

Thinking rationally

The approach is driven by logic type of reasoning. Artificial Intelligence designed on this philosophy aims at solving problems using a logical approach, implementing solutions that aims at decomposing and solving the problems using logical reasoning. This approach has two drawbacks:

- The link between informal knowledge and formal representation by logical notation is not always easy to obtain and define
- The number of steps to be taken by a computer problem is not directly related to the execution time and to the computational resources needed, as even simple problems can hinder the computational resources if no guidance is provided to identify the correct initial actions to take

Acting rationally

Computer programs developed with this philosophy are expected to operate without external control, sense their surroundings, adapt and create and follow goals. A rational agent constantly aims at achieving the best results, or, in case of uncertain conditions, the best expected outcome. In general, rational agents will tend to execute actions defined by either inferences or other kinds of reasoning. Among the qualities needed for an agent to act rationally one can include those needed to successfully pass the Turing Test, knowledge representation and reasoning. When compared to the other approaches, two better qualities characterize the rational agent approach:

- Generality: this approach encompasses more possible mechanisms to achieve rationality with respect to the "laws of thought"
- More compliant with scientific procedures and development, as the approach is formally defined and completely general

5.2 Brief history of Artificial Intelligence

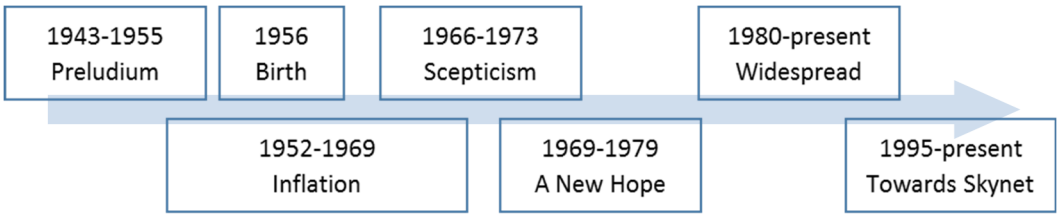


Figure 16: History of Artificial Intelligence

The rise of AI as a distinct field in CS was not characterized by a linear progression: instead, after an initial positive reception by the scientific community, the field had to face several problems that highlighted the limitations of both the State of the Art (SoA) algorithms and of the computer machines that were available at the time. Despite this tormented start, AI is now unmistakeably recognized as one of the most prominent fields in CS: it is therefore useful to recall all the key steps in the evolution of the field.

1943-1955, the Preludium

The two professional figures widely recognized as the fathers of AI were the neuroscientist Warrant McCulloch and the mathematician Walter Pitts, with a work that is now indicated as the ancestor of AI: the proposal of a model of artificial neurons in which each unit can be in the states of "on" and "off", where the "on" state occurs after a sufficiently strong stimulation by the neighbour neurons. The results come after considering three main contributions: the fundamentals of physiology and the study of the functions of neurons in the brain; Russel's and Whitehead's formal analysis of propositional logic and Turing's theory of computation. Further works by McCulloch and Pitts on the field of network learning introduced a simple rule to update the connections between neurons, the Hebbian learning, which has been a pioneering model for years. The exemplar work in the early developments of AI were made by Alan Turing, that introduced the Turing Test, machine learning, genetic algorithms, and reinforcement learning concepts.

1956, the Birth

The key event in the history of AI can be identified in the workshop organized in Dartmouth in the summer of 1956, in which 10 selected researchers participated in two months period of research on the topics of AI. In this workshop, embryo applications were developed and presented, including what can be considered the first reasoning program capable of thinking non-numerically. Although the workshop itself did not hold significant progresses in the field of AI, it served as a fruitful start of the collaborations that led the AI development scene in the following two decades. Starting from this event, two key and distinct characteristics of AI development were made evident: the aim of AI researchers of duplicating human faculties such as creativity, self-improvement and language use; and the research on methodologies that focus on building machines that will function autonomously in complex, changing environments.

1952-1969, the Inflation

The era of computers was at its beginnings, machines and programming tools were still limited and the functions they could perform were basic, especially in the earlier years of this period. Nonetheless, AI researchers were constantly confronted with the idea that computers could never be programmed to do certain tasks. One after another, the researchers could implement most challenges that were posed in those years. Interesting applications of those years were the General Problem Solver, a program designed to implement the “thinking humanly” approach, that could solve problems in a way similar to that used by humans. Applications for playing checkers were also developed using AI. Key advancement in the programming tools available at the time was the invention of the language Lisp, that will be the leading programming language for AI for the next 30 years. Initial demonstration of self-learning programs was also realized during this period. Concepts such as Adalines neural networks and perceptrons were also introduced.

1966-1973, the Scepticism

Initial successes came abundant as the AI research carried on. Despite this promising evolution, the development of AI-based applications soon encountered key issues that characterized those years: AI systems were performing very well in specific but rather simple examples, while they would fail poorly when tested on wider or more complex problems. In particular, a very challenging aspect was the fact that most of early AI programs would fail in solving problems they knew nothing about. One key example can be traced to machine translation efforts, that showed how knowledge and understanding of the speech context is mandatory to

perform an accurate translation, and implementing those traits in a translator program turned out to be more difficult than what had been predicted. Another misconception that arose in those years was based on the fact that problem size was irrelevant: a program able to solve a small but generic problem, was thought to be able to solve more extended and vast problems as well, the difference being only in the hardware that was running the algorithm. Scaling up to larger problems was believed to be only an issue of more performing hardware and memories. Furthermore, intrinsic limitations on the basic structures of earliest AI development limited the performances of those algorithms.

1969-1979, a New Hope

The research carried on in the previous years was focused on what have been called "weak methods", as, despite being general, they do not show the same type of performances with smaller or bigger problems, therefore showing scaling issues. The solution to this issue was found to implement more powerful, specific methods that allow more versatile reasoning. A few examples appeared in this period, such as the DENDRAL program, that was able to determine the molecular structure only by considering the outputs of a mass spectrometer. This program represented the first knowledge-intensive system: its behaviour was originated from a large quantity of special-purpose rules. Another notable project has been HPP, the Heuristic Programming Project that was exploring the feasibility of expert systems and their application in other fields of human expertise. In this sense, the research was directed towards medicine and diagnosis. With a program made of about 450 rules, the performances of this expert system could be compared to those of an expert physician, while reliably being better of a junior doctor. Notable at this time was also the introduction of uncertainty during problems solving.

1980-present, the Widespread Adoption

The adoption of AI algorithms by companies worldwide saw both promising and cautious times: the years after 1980 were surely considered a positive period for AI applications: these allowed several companies to save great amounts of capitals, and each of the leading CS companies had their own AI research team, with a consequent investment in AI industry that rose to reach billions of dollars by 1988. Examples of this effort were realized in the field of expert systems, vision systems, robots and specialized hardware and software.

At the same time, the back-propagation algorithm (invented in 1969 by Bryson and Ho) came back in fashion and was applied to many learning problems, from computer science to psychology. Several interesting results were obtained thanks

to this training algorithm, and these successes contributed to create the third distinct approach to the study and development of AI applications.

At last, the evolution of the research around AI separated into two distinct efforts: researching on effective network architectures and algorithms, and research on reaching precise modelling of the biological neurons and their group architectures.

The latest direction of development of AI are towards an embrace of the scientific methodology that is the standard in other research fields. AI research must now undergo rigorous empirical experiments, and the results must be analysed statistically for their importance. Shared repositories of test data and code made it possible to replicate experiments with ease.

This, coupled with refinements on the tools available to the AI researchers (such as the Bayesian networks and improved training algorithms) allowed AI algorithms to reach significant results in fields traditionally dominated by statistics, pattern recognition, machine learning and so on.

1995-present, towards Skynet

Huge successes in the various fields of AI have contributed to the affirmation of this branch of CS. Despite these successes, in the latest years, a particular research effort has taken back momentum and is now expanding: the strive towards the "whole agent". Furthermore, previously isolated fields of AI have now been joined together, comparing and sharing each other's results: it is a fact that sensory systems (vision, sonar and speech recognition) cannot deliver reliable information about the environment. For this reason, reasoning and planning systems must be able to handle uncertainty. In addition, another consequence of the agent perspective is that AI has been drawn into much closer contact with other fields, such as control theory and economics, that also deal with agents.

More exotic research directions (that, on the other side, share similar intents with initial efforts in AI research) are considering the emulation of the human-level intelligence, or more in general the development of an Artificial General Intelligence, that would implement an universal algorithm for learning and acting in any environment.

Finally, in recent years, an important paradigm shift has begun to appear: thanks to the increased availability of data, scientists and researchers are becoming less picky about the choice of the algorithm, with respect to careful definition and construction of the datasets involved in the application. Examples of this can be found in

linguistic (trillions of English words), in pattern recognition (billions of images found in the internet), in genetics (billions of base pairs of genomic sequences).

Researches like these highlight the possibility that the current problem of AI is the way all the knowledge needed in an application is expressed, and that this problem can be solved by improving the data used and then the learning method used, rather than hand-crafting the knowledge into the problem. This type of approach is suitable in different fields, but holds less value in field of application where the datasets available are more limited in size.

5.3 The basis of Artificial Intelligence

The definition and establishment of Artificial Intelligence as a prominent field in Computer Science is the result of an evolution of ideas, viewpoints and methodologies that started out earlier in the human history with respect to the invention of computers. In general, the path that led to the definition of Artificial Intelligence as a discipline can be described under several different lights: in any case, four distinct incentives and pushes can be identified in the areas of philosophy, mathematics, economics and neuro-science.

Table 9: Foundations of Artificial Intelligence

Philosophy	**Mathematics**
Earliest records of automating human reasoning date back to Aristotle (III B.C. century) that formulated a methodology to rule the rational sphere of the mind. He developed an informal system of syllogisms for proper reasoning, which allowed to obtain definitive conclusions given initial premises. After him, the mechanization of the thinking act was explored by Lull (XIV century), that envisioned reasoning carried out by a mechanical artefact; Hobbes (XVI century) proposed to treat reasoning as numerical computation. Continuing,	The advance in mathematics was one of the key stepping stones of the definition of the foundations of AI. In particular, the three essential fields that can be linked to AI are logic, computation and probability. Each one of these fields has its own origins and main exponents (Boole, Frege, Tarski). The development and definition of the first algorithms (Euclid, III B.C. century) is a citation that has to be done as well as the first works on proving which mathematical problems could or could not be proved (Gödel, XX century) and more generally the effort

several philosophical currents can be linked to the origins of AI, among which are: rationalism, dualism, materialism, empiricism, and so on. In general, the typical questions of the philosophic effort can be identified in:
- do formal rules to obtain valid conclusions exist?
- what constitutes the mind and the physical brain?
- where does knowledge come from?
- how is knowledge translated into action?

to characterize which functions are computable (Turing, XX century). Moreover, tractability problems and NP-completeness are surely subjects that are involved with the development of AI. Relevant questions:
- Are there, if any, formal rules to draw valid conclusions?
- What can and what cannot be computed?
- How do we deal with uncertain information?

Economics

The field of economics is a relatively recent one when compared with philosophy and mathematics, yet it held very important results that fostered the development of the AI discipline: the mathematical theory of preferred outcomes (or utility); the decision theory, that combines probability theory with utility theory and later on the game theory. Of paramount importance are also the operation research and the Markov decision processes.
Some of the fundamental questions of the field, related to AI:
- how should we make decisions to improve the outcomes?
- how can we change these decisions when the outcomes are evaluated in the far future, or when boundary conditions vary?

Neuroscience

Neuroscience is involved with studying the brain, which is the main element of the nervous systems in human beings. Despite the majority of the brain's characteristics and functions are yet to be discovered, several advancements were made in the study of localized areas of the brain responsible for specific cognitive functions (Broca, XIX century). On the other side, the study of the brain nerve cells, the neurons, was carried out after a staining technique was invented that allowed the observation of individual neurons in the brain (Golgi, XIX century). Furthermore, after the invention of the electroencephalograph, the measurement of intact brain activity could begin. Recent developments of functional magnetic resonance imaging are providing neuroscientists with

incredibly detailed images of the brain activity. One of the most promising conclusions of this discipline is that a collection of simple cells can lead to thought, action and consciousness (Searle, XX century). The leading question that is having its effects on AI development is:
- how do brains process information?

Psychology

The early advancements on experimental psychology began with rigorously controlled experiments on human beings (Helmholtz, Wundt, XX century). Another effort was led by the behaviourism movement, that aimed to study only objective measures of the stimuli given to an animal and the resulting actions. The definition of the brain as an information-processing device, and the involvement of the perception as a form of unconscious logical inference can be traced back to the end of XIX century. Further developments were made towards the definition of what is known as a knowledge-based agent, which possesses three characteristic traits: a stimulus is translated into an internal representation, the representation is processed by cognitive functions to derive new internal representations, and these are translated back into action. The leading question related to AI in the field of psychology is:

Computer Engineering

The missing piece so far in the development and spreading of AI is a type of technology that allows the implementation of the AI algorithms. The selected choice has obviously been the computer, despite calculating devices were invented before the computer, but were overcome by the adoption of the computer. On the other side, the software side of computer science, several developments were essential for the diffusion of AI: programming languages, operating systems and tools.

The main driver in this area has been:
- how can we build an efficient computer?

- how do humans and animals think and act?

Control theory and cybernetics

Several examples of early control theory are spread throughout history, starting from water clocks with regulators (II B.C. century), to self-regulating feedback control systems (steam engines governor, Watt, XIX century). Control theory has been introduced by Wiener, that also speculated on creating artificially intelligent machines by the use of homeostatic devices, implementing appropriate feedback loops to achieve stable adaptive behaviour. Latest development in control theory have all aimed at reaching the maximization of an objective function over time (see stochastic optimal control). For this reason, the advancements in control theory can often times be placed side by side with advancements in AI. Calculus and matrix algebra, the tools of control theory, lend themselves to systems that are describable by sets of variables, whereas AI was founded in part as a way to escape from these perceived limitations. The tools of logical inference and computation allowed AI researchers to consider problems such as language, vision and planning that fell completely outside the control theorist's view. Leading research vision:

Linguistics

Linguistics also played a major role in the development of AI, mostly because it provided the missing link between human language and computers, with theories such as computational linguistics or natural language processing and knowledge representation.

The understanding of human language turned out to be a joint effort between understanding the subject matter, the context and the structure of sentences. Furthermore, the link between language and thought has been considered very important:
- How does language relate to thought?

- how can artefacts operate under their
own control?

5.4 State of the Art

Identifying the SoA for the broad field of AI is certainly not a trivial task, and the search must take into account the speed with which these algorithms and their applications are evolving.

5.4.1 What belongs to Artificial Intelligence

One of the most striking characteristics of the field of AI is the ever-present evolution in the algorithms and applications that can be considered part of the field. In this section, a summary of the major algorithms of Artificial Intelligence are presented. In general, when a research field is so vast and with so many different applications, it's difficult to include all the known algorithms in a concise summary. The idea is to describe the constellation of the elements in this research field by highlighting first the different algorithms and how they are grouped, and then by citing the most promising and interesting applications that are solved with the use of AI.

5.4.2 State of the Art by algorithm

Problem-solving

The category of algorithms that have the purpose of solving problems can be grouped together, as they represent a set of general-purpose algorithms that search for a solution to problems that, in this case, have as solutions a fixed sequence of actions: in general, the representation of the problem could involve branching in order to recommend different actions depending on the situation.

Solving problems by searching

This category of algorithms includes the searching strategies: defining the different methods to explore and move through a tree that represents and describe the problem itself. Examples of algorithms in this category include the *breadth-first search*, where all the nodes at a given depth in the search tree are expanded before any nodes in the next level are. Drawbacks of these methods are that memory

requirements are usually a great concern and execution times are often not practical. On a similar note, *depth-first search* suffers from similar issues and are both not optimal search methods. A decent solution is represented by *iterative deepening search*, which tries to combine the benefits of breadth- and depth-first searches: this method is the preferred uninformed search method when the search space is large and the depth of the solution is not known. More exotic searching is represented by the *bidirectional search*, where two searches are performed, one from the root node and one backwards from the goal. An improvement over uninformed search is to perform *informed searches*, when possible. The improvement comes from the fact that evaluating the current state allows to introduce efficiency in the exploration: *best-first search* is one example, *greedy* variant introduces a choice based on preferring the expansion of the node closest to the goal, considering that that node will be the most likely to lead to a solution. The currently most widely known form of best-first search is the *A* search*, that combines the information of the cost to reach the node and the cost to get from the node to the goal. Memory bounded versions of the introduced algorithms exists as well (*recursive best-first search* and *simplified memory-bounded A**).

Beyond classical search

An evolution and a differentiation with respect to traditional search models is represented by the category of algorithms that do not implement systematic searches of all the possible paths. They have two key advantages: small amount of memory and perform quite well in large or infinite state spaces. *Local search* algorithms are an example, and are useful for solving optimization problem where the intent is to obtain the best state given by optimizing an objective function. Traditionally these algorithms involve analysing the shape of the objective functions to find the global minimum (or maximum), avoiding local extremes and plateaux, and coping with ridges (which are traditionally very difficult to deal with for local search algorithms). *Hill climbing* and its variations are an example. To overcome the problems of avoiding local extremes, an algorithm that combines the processes of *Hill climbing* and the exploration properties of a random walk is *Simulated annealing*, that introduces the concept of temperature while performing gradient descent. *Beam searches* introduces the characteristics of exploring more than one generated state, while keeping a connection between the searches, and passing useful information between them.

A particular case of *stochastic beam search* is defined by *Genetic Algorithms*: they aim at emulating the dynamics of populations of individuals, implementing the

survival of the fittest law of nature. In particular, the search for an optimal solution is done by encoding the single solution as a single individual: it will then evolve in successive generations of the population, converging to the optimal solution. Behaviours such as reproduction, mutations, parenthood, natural selection and elitism are defined and are essential for the success of the algorithm.

A note to these algorithms: it must be said that the strategies can vary when we deal with problems in which the agent possesses sensors, and the strategies are different in the case of a fully observable world, a partially observable one, and a non-observable one.

Adversarial search

One of the key characteristics of *Adversarial Search* problems is that they deal with competitive environments, such as games. Most of the times, real-life games are quite difficult, if not impossible, to solve completely. One of the most important reasons is because of the dimension of the problem. The average branching factor of chess is 35, with games that can reach 50 moves per player. In such cases, defining the optimal move is unfeasible. Several techniques exist to facilitate the decision during games, such as pruning, that allows the algorithm to ignore portions of the search tree, evaluation functions to approximate the true utility of a state without a complete search, and strategies to deal with imperfect information. Famous algorithms in this case are minimax for decision making, alpha-beta pruning for removing large parts of a search tree, and in some cases, table lookup for games states which solutions are known a-priori thanks to human knowledge and experience. Even in this case, distinctions are possible when we consider games ruled by chance or not, and games where the information is perfect or imperfect.

Constraint satisfaction problem

A more efficient approach for solving specific problems is known as *Constraint Satisfaction Problem (CSP)* and involves a type of problems that is defined by setting constraints to the characteristic variables, and that is solved when each variable has a value that satisfies all the constraints on the variables. With respect to traditional state-space search, the algorithms that solves CSPs involve two possible actions: *search*, similar to traditional state-space problems, and do a specific type of *inference*, that is propagate the constraints: this means exploiting the constraints to reduce the number of values that a specific variable can assume. Several types of inference techniques exist to check the consistency of the

constraints, common ones being *node, arc, path and k-consistency*. Searches can be performed tracking backwards (*backtracking*) and methods exist to choose the best variable to explore during a backtracking search.

Knowledge, reasoning and planning

The group of algorithms that aim to solve problems by using reasoning on an internal representation of knowledge is constituted by *knowledge-based agents*. This type of agents represents an evolution to what described earlier, as they operate exploiting *logic*, seen as a general class of representations to support knowledge-based agents.

Logical agents and First-order logic

Logical agents are a category of agents that use formal logic to take decisions and perform actions in their world. Logic is the key element in the behaviour of the agent, and is characterized by the presence of a syntax, semantics, knowledge-base, and an inference procedure.

First-order logic is a type of logic that is inherently more powerful than propositional logic. In this case, the types of problems that can be solved are more complex and can be solved more efficiently with respect to propositional logic. In general, developing a knowledge base in first-order logic requires a careful process of analysing the domain, choosing a vocabulary, and encoding the axioms required to support the desired inferences.

Classical planning and complex planning

Planning systems are problem-solving algorithms that operate on explicit propositional or relational representations of states and actions. One of the most famous algorithms is PDDL, the *Planning Domain Definition Language*, that describes the initial and goal states as conjunctions of literals, and actions in terms of their preconditions and effects. Planning graphs are often used to contain supersets of all the literals or actions that could occur, and yield useful heuristics for state-space and partial-order planners. There is no consensus on which planning algorithm is currently the best, yet cross-fertilization between algorithms has yield successful progresses.

Complex planning introduces the concepts of scheduling and of resource constraints, which are not considered in classical planning. Strategies such as *hierarchical task network* planning are used to infuse advice in the agent by domain

designers in the form of high-level actions. Extensions of this theory comprise online planning and multi-agent planning.

Uncertain knowledge and reasoning

Dealing with uncertainty is one of the pillars of modern agent design: an agent needs to handle uncertainty, both in case of partial observability, or non-determinism, or a combination of the two. When we analyse the approaches of agents described earlier, a few drawbacks are highlighted: when a logical agent has to consider every logically possible explanation for the observations, the belief-state representations become large and complex; on the same side, a correct contingent plan that handles every eventuality can grow large and is daunted by several low-probability events; when no plan is definable to pursue a goal, an action is still required to the agent.

Dealing with Uncertainty

When dealing with uncertain reasoning, especially in complex problems, it is mandatory to define a way to quantify the uncertainty level. For this, probability comes into play with a way of summarizing the uncertainty level. Once a description of the uncertainty is obtained, the subsequent actions consist in defining agent preferences (what to do with respect to probability) and which utility is reached once a preferred action is taken. The fundamental idea of decision theory is that "*an agent is rational if and only if it chooses the action that yields the highest expected utility, averaged over all the possible outcomes of the action"* (Maximum Expected Utility, MEU). Bayesian Networks represent a very effective way to represent uncertain knowledge. They are a directed acyclic graph whose nodes correspond to random variables, with each node storing a conditional distribution for the node. Inference algorithms exist to calculate the probability distribution of a set of query variables, given a set of evidence variables.

Introducing the concept of time when dealing with uncertain reasoning requires the introduction of more versatile reasoning tool, that have been widely used in the last decades: the Markov processes and models. Inference models need to be updated to take into account the dynamic environments. Powerful algorithms to consider are also *Kalman Filters, Dynamic Bayesian Networks, particle filtering* and more.

Decision making, simple and complex cases

Decision making is a process that combines having information of the environment (thanks to probability representation) and information on the obtained utility of a certain action (thanks to utility theory). In general, different types of tools are available, but most promising ones are *Multi-Attribute Utility Theory* and *Decision Networks.* Finally, *Expert Systems* include utility information and have additional capabilities when compared with pure inference systems. On the other hand, the problem of decision making is much more complex when we deal with sequential decision problems, that are a category of problem in which the agent's utility depends not only on the outcome of a single decision, but on the sequential outcomes of more decision actions. When the problem can be described as a sequential decision problem for a completely observable, stochastic environment with a Markovian transition model and additive rewards is called *Markov Decision Process (MDP)* and they became one of the most important algorithms to date. A variation to MDPs occurs when the environment is not fully observable, thus encountering the *Partially Observable MDPs.*

Learning

The concept of learning is one of the fundamental aspects of AI, and defines a category of algorithms that are known as *Machine Learning*. Learning means that the performance of an agent will improve on future actions after observing the surrounding world. There are three key reasons for why a developer would prefer learning algorithms over hard-coded software: first, not every situation the agent will be in can be predicted by the designer; second, changes over time are difficult to predict; third, for certain problems, the direct implementation of a solver algorithm is too hard and automatic learning represent the only viable solution to the implementation of an agent. In general, four topics are shared among different learning algorithms and problems: there is a component of an algorithm to be improved; the agent possesses prior knowledge; data is represented in a specific way; a feedback action provides guidance during learning. When a specific algorithm needs to learn from its surrounding world, three main learning algorithms are available to the designer, and will be discussed later: reinforcement learning, supervised learning, unsupervised learning.

Learning from examples

Among the algorithms that can learn from examples, *decision trees* can be cited. When learning from examples, one question appears soon: when an algorithm

learns something, how can we be sure that our learning algorithm has produced a hypothesis that will predict the correct value for previously unseen inputs? How many elements in the dataset do we need to get a good hypothesis? What hypothesis space should we use? An interesting principle of computational learning theory states as follows: "*any hypothesis that is seriously wrong will almost certainly be found out with high probability after a small number of examples, because it will make an incorrect prediction. Thus, any hypothesis that is consistent with a sufficiently large set of training examples is unlikely to be seriously wrong: that is, it must be probably approximately correct*". Algorithms built on this principle are called *Probably Approximately Correct Learning (PAC-learning)*. Classification algorithms (linear, linear with regressions, linear with hard threshold, and so on) can be considered.

An exemplar category of algorithms that exhibit learning characteristics are the *Artificial Neural Networks (ANN)*. The algorithm aims at representing the functional behaviour of biological neurons, translating their functions into computational units, the artificial neurons. A simple mathematical model for a neuron will be described later. When the artificial neurons are grouped together, a network is defined. Several types of networks exist, such as feed-forward, recurrent, single- or multi- layer, networks, and so on. Learning, which will be dealt with later, is performed by applying training algorithms on a dataset.

Several other algorithms exist: *Nearest Neighbour Models, Support Vector Machines, Ensemble Learning*.

Several other methods consider prior knowledge during learning: in this case, the effects of knowledge representation and learning are joined together. *Current-best-hypothesis search* and *Least-commitment search* are examples of these algorithms. Efforts are also being spent in developing methodologies to extract general knowledge from specific examples. Several different types of learning have been developed, including *Explanation-based learning, Relevance-based learning, Knowledge-based inductive learning, Inductive logic programming*.

Learning probabilistic methods

Learning by using statistical methods can be done in several different ways, and the methods that fall under this category can be theoretically simple or very complex. A few examples are: *Bayesian learning, Maximum a posteriori,*

Maximum-likelihood and by using *non-parametric models*. The majority of the algorithms in this category fall under the unsupervised learning strategies.

Reinforcement learning

The fundamental concept behind reinforcement learning is that, providing the agent with a feedback about how good (or bad) were the decisions he took, will eventually improve its decision-making capabilities. This type of feedback is identified as a reward, or reinforcement, and can be given either at the end of a series of actions or more frequently. Two main philosophies exist when considering reinforcement learning, passive and active reinforcement: in the first, the agent's objective is to compute each states' utility, while in the latter the agent must determine which actions to take. In general, anyway, the methodology used to design an agent is tied to the information that needs to be learned. In the case of the passive reinforcement learning, the utilities can be computed by using *Direct utility estimation, Adaptive dynamic programming, Temporal-difference.*

Communicating, perceiving, and acting

When dealing with the problem of acquiring knowledge, the most powerful ability that an agent can possess is the ability to understand natural language, as the majority of the information currently stored in computer is expressed in this form of language.

Understanding natural language

In this sense, *n-gram models* represent a quite effective methodology to represent and learn the letter and words in a natural language. Smoothing of the n-gram models is a process that allows to avoid the limitations of the training dataset: above all, the backoff model is one of the better performing.

Machine reading holds a predominant spot in this section: the intent is to build a system to extract knowledge from written text that can work with no human input of any kind: a system that could define and fill in its database. In general, it is necessary to define not only a system to parse and grasp the knowledge, but also to explore the actual human behaviour. Several algorithms have been used so far for different problems related to this problem: *treebank* is useful to learn a new grammar, *CYK algorithm* can learn sentences in a context-free language, *lexicalized PCFG* allows us to represent connections between words that are more frequent wrt

others. The same algorithms and concepts can also be applied to speech recognition problems.

Perception

Perception is a fundamental field for computer science and in particular for applications that interact with an environment, be it the real one or a virtual one. The basic concept behind perception is that a device, known as *sensor*, execute a measurement of the surrounding environment and provides it as an input to an agent. Sensor is used here as a broad term, not linking the meaning to a complexity threshold, but to the fact that any information is collected, manipulated and shared. By intrinsic nature, a sensor that observes the real world will create a *distorted* perception of the environment. This fact could be false when considering virtual sensors. Once the external world is sampled, it is necessary to perform post-processing to extract meaningful information: the type of algorithms used at this point can vary, for example in the case of object recognition algorithms such as *Scale Invariant Feature Transform, Histogram of Gradient orientations,* and *Neural Networks* can be used. Algorithms might increase in complexity when specific portions of an object need to be characterized as well, for examples arms and legs in the human body.

Complex systems

When the problem to be solved involves the development of a system that actively performs actions, we are dealing with complex systems, of which *robotics* is a category. Traditionally, the base definition of a complex system is a system that possesses three characteristic traits: sensors, actuators and a brain, or controller. The diversification of the complex systems is astonishing: manipulators, humanoid robots, UAVs and planes, spacecraft.

5.4.3 State of the Art by application

A small set of applications that notoriously implement some form of AI are presented here. As for the previous list, a comprehensive list is difficult to produce and would become outdated very quickly in this ever-changing environment. Nonetheless, it is interesting to recap some of the main contribution of AI in key example applications.

Autonomous planning and scheduling – Space Agencies (NASA, ESA, JAXA) have since long dealt with autonomy and the problem of enabling the

spacecraft (especially interplanetary probes) with decision-making capabilities. In the last two decades, a couple of applications were embedded on NASA spacecraft: REMOTE AGENT featured autonomous execution plans, deriving them from high-level goals sent from ground. NASA invested further research on goal-generation capabilities, developing CASPER, which enabled the spacecraft with decision making capabilities, goal generation, real-time scheduling, repairing and optimization [38].

Fault Detection, Isolation and Recovery – Fault Detection systems have been developed using several categories of AI algorithms, ranging from model-based applications (which can be considered on the border of AI), to Fuzzy Logics and Neural Networks [48].

Game playing – The art of gaming has always been a field where AI research has been focused on, since the earliest decades of the diffusion of these algorithms. Traditionally, each game saw the development of a specific tailored algorithm, and the common long-term goal has always been to challenge and beat the world top players in each discipline. IBM's DEEP BLUE has set a keystone event in the game of chess, beating world champion Garry Kasparov in an exhibition match. Games such as Scrabble, Go and Jeopardy all saw the top players being beaten by AI in the following years, with Go and Jeopardy games being one of the most challenging efforts because of game complexity and size of possibilities during the game.

Logistic planning – AI applications for logistics planning and scheduling in transportation have been developed both for civilian and military cases, with an emblematic case being the use of the Dynamic Analysis and Replanning Tool, used by DARPA to plan starting points, destinations, routes and conflict resolutions of people and cargo. The improvement in the definition of these plans was so significant that the time for the generation of these plans was reduced from weeks to hours, with incredible increase of savings.

Machine translation – Language translation has seen a dramatic improvement in the quality of the translations after specific AI algorithms have been developed and used. Teams of researchers are able to develop high-performance translators just by having deep notions of statistics and machine learning algorithms, without knowing the languages themselves. The program uses a statistical model built from example translations and from examples of texts totalling trillions of words. Moreover, applications such as Skype Translator or Google Translate are making extensive use of AI algorithms and the results are outstanding.

Medical research – Machine Learning, Markovian Decision Processes, Expert Systems are only a few examples of performing algorithms in the field of medicine: they are used to implement "whole agents", such as Watson from IBM, or to solve specific problems such as cancer/gene researches, image changes assessment, and more [49].

Robotic vehicles – The SoA for civilian robotic vehicles (cars, trucks) has considerably improved in the last decade. Several car manufacturers are now testing their autonomous vehicles on roads open to normal civilian traffic (Tesla, Google, Volvo cars, Scania trucks) [50]–[54]. Concerning non-civilian robotic vehicles, companies are developing interesting applications for quadruped robots (Boston Dynamics, DARPA). Excellent examples of applications are also to be found in interplanetary robotics systems, such as NASA Mars Science Laboratory [55].

Robotics – Research applications have differentiated into various fields, encompassing aerial, terrestrial and underwater robots: examples are found in the heavy industry, in paralyzed people aids, computer vision and so on. Other applications involved are the Touring Problems, VLSI layouts, Automatic Assembly Sequencing and so on.

Spam fighting - when dealing with online spammers, a static programming approach is not flexible and agile enough to keep pace with the evolution of the spammers algorithms.

Speech recognition – current advancements in speech recognition are proven by the fact that AI-enabled computers are now performing better than humans in recognizing the words in a speech.

5.4.4 State of the Art by Open Source products

Open Source approach has become nowadays a fundamental key to the advancement of international research, as open source programs and libraries allows the researchers to focus on their research application, with respect to focus on the development of the learning technology to be used.

Python-based open source, deep learning tools

TensorFlow – one of the most performant machine intelligence software libraries available. Developed by Google engineers and researchers, is used for numerical computation using data flow graphs.

Pylearn2 – a machine learning research library. Designed to make machine learning research easy, development status is on-hold.

Theano – Python library to design, optimize and evaluate mathematical expressions involving multi-dimensional arrays. Supports several frameworks.

Blocks – Theano framework to build and train neural networks.

Lasagne – Another famous Theano framework to build and train neural networks.

Matlab-based open-source, deep learning tools

DeepLearnToolbox – Matlab toolbox for Deep Learning.

Deep Belief Networks – Matlab code for training Deep Belief Networks.

Deepmat – Matlab based deep learning algorithms

MatConvNet – Matlab toolbox to implement Convolutional Neural Networks (CNNs) for computer vision applications.

Matlab Deep Learning – standard Matlab documentation on deep learning.

C/C++-based open-source, deep learning tools

CUV library – C++ framework with python bindings for easy use of Nvidia CUDA functions on matrices.

OpenNN – An open source class library written in C++, which implements neural networks

Eblearn – C++ machine learning library for energy-based learning, convolutional networks, vision/recognition applications.

CXXNET – Fast, efficient and lightweight C++/CUDA framework with friendly to python/Matlab interface for training and prediction.

The software list is not comprehensive of all the development and product efforts in the different available languages. Updated information can be found at [56].

5.5 Bringing Artificial Intelligence to space

5.5.1 Selection of CubeSat compatible algorithms

Artificial Intelligence categories, algorithms and disciplines are numerous, and several approaches could be used to tackle specific problems. One of the questions that arises in this situation is how do we choose among the variety of available algorithms? Is there some kind of preliminary cut-off that severely limits the applicability of certain algorithms in space missions, and in particular in small satellite ones? Is this cut-off applying to CubeSat-vs-Traditional platforms or to Spacecraft-vs-Ground categories? One of the striking features of CubeSats platforms is the extensive use of COTS. Among these, the selection of computing units available is united by a peculiar characteristic: the average performance of COTS processors is considerably higher than the average rad-hard solution found in spacecraft. However, despite the average performances, interesting solutions from the rad-hard domain are also appearing, and this makes the separation between COTS and rad-hard processors somewhat smaller when interpreted from a performance point of view [57]. Even if from a cost perspective the comparison might still be unfair, the performance difference, at a first glance, does not seem to drive the selection of available algorithms. Another comparison that can be made is between the average computing power available on CubeSats and the average computing power of ground-based systems. Ground-based computers, especially those traditionally used for Artificial Intelligence algorithm development, benefit of elements that are not included in spacecraft computing units: Graphics Processing Units (GPU). These types of computing units are designed to perform intensive jobs, exploiting the parallelism in their architecture, that allows to optimize the workload in a great number of parallel threads. These type of devices, when compared to CPUs, trade a vastly improved computational power for demanding tasks with a greatly increased power consumption. AI applications developed on ground make use of clusters of GPUs, which is obviously not achievable on a spacecraft. This is most likely the first cut-off concerning the usage of AI algorithms but in principle this cut-off does not exclude certain algorithms from being applied: the result is that, in order to apply algorithms that are power-intensive on ground, a modification in their architecture must be envisioned for the space segment. Examples of these modification can be a reduction in size of a Neural Network, or an optimization of the training dataset used by the application.

5.5.2 Mapping Artificial Intelligence algorithms to fields of application

Given a particular problem to solve, or an application to develop, several different algorithms that are considered AI could be applied, and the solutions obtained with these different algorithms would likely be similar, or at least comparable. In fact, striking distinctions in terms of performances, computational cost and other parameters (very important when applying the technology to a space mission) would likely be discovered and evaluated later in the process of exploring the feasibility of applying a specific algorithm to the problem. A mapping between the three applications presented in the thesis and potential AI algorithms that could provide a solution to it are shown in Figure 17.

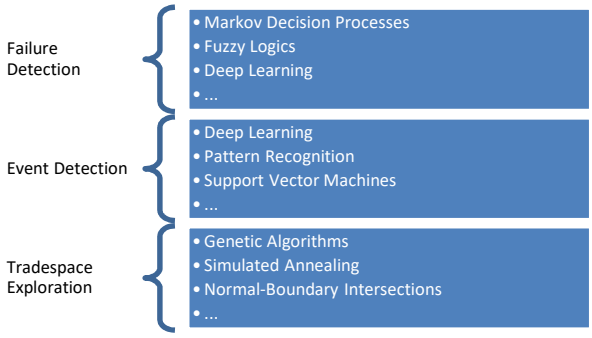


Figure 17 Mapping between applications presented in the thesis and potential Artificial Intelligence algorithms to solve those problems

Given the enormous availability of algorithms, the mapping does not aim at being exhaustive: provides a small view on known algorithms that are most likely to return interesting results and performances.

5.6 Machine Learning algorithms and Neural Networks

Definitions of Machine Learning started as early as 1959, with Arthur Samuel defining ML as:

"field of study that gives computers the ability to learn without being explicitly programmed"

This is the prelude to an incredibly vast, and ever growing, world [58]. Figure 18 attempts at presenting an overview of all the algorithms that can be qualified as ML. As any type of mapping of a complex world, there are some imperfections.

Some methods could be represented by more than one category, for example, but in general, grouping the different methods by similarity in terms of functionality is one of the most effective approaches. The map presented is not meant to be exhaustive.

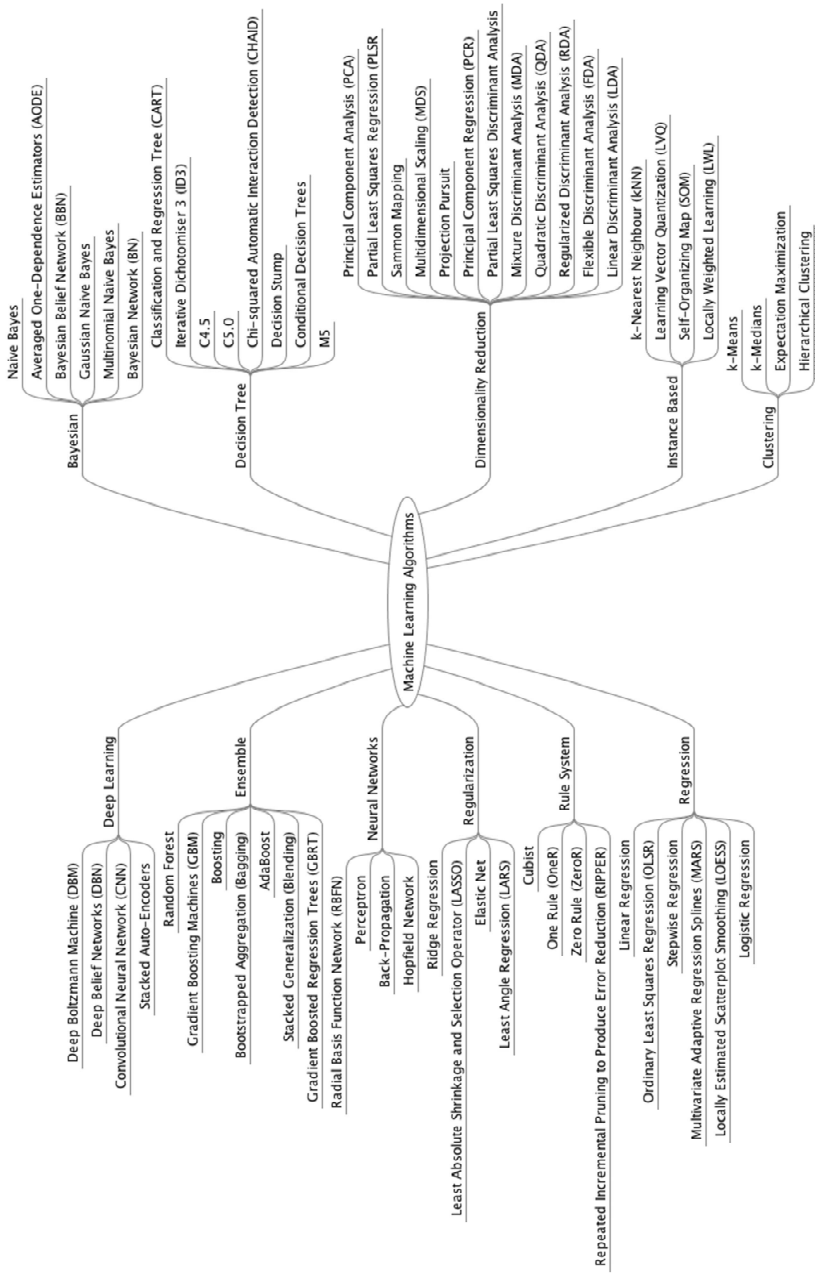


Figure 18 Machine Learning algorithm map, grouped by type. Credits Brownlee.

As it can be imaged, such an enormous availability of algorithms that fall under the category of ML, implies that peculiar problems encountered during a space mission, such as event detection, image classification or mission replanning could be solved by applying many, or many combinations, of the ML algorithms presented in the image.

5.6.1 Neural Networks Principles

The chosen family of algorithm to perform event detection on a spacecraft has been Artificial Neural Networks. Before digging into the characteristics and different types of algorithms that fall under the ANN category, it is important to state some of the characteristics that made ANN a good candidate for this type of problems:

- Generalization: a trained network can provide good results even on never-before-seen inputs, provided that they are similar to those the network has been trained on
- Experience: a network, similarly to human behaviours, is able to learn thanks to the knowledge that is fed into it
- Ability to deal with linear and non-linear functions, and has multi-variable capabilities
- Robustness in presence of noise, disturbances and degradation. Generally, the performance of a network degrades gracefully under adverse operating conditions
- Performances can be better than a human counterpart, even if the knowledge with which the network is trained comes from the human expert

As with other types of AI, training and execution of ANN does not follow traditional approaches, and the definition of the application behaviour is not implemented through conventional programming.

ANNs have been introduced with the intent of modelling the processing capabilities of biological nervous systems: millions of interconnected cells, each one of them being a complex machine in which incoming signals are collected, processed and routed in several directions (the neuron). From a computational speed point of view, the common neuron is thousands of times slower than our state of the art electronic logic gate: despite this, the human brain is able to achieve complexity of problem solving that is yet unmatched by computers.

5.6.1.1 Biological Model

There are several differences between the biological neuron and the computing unit known as neuron used in ANN.

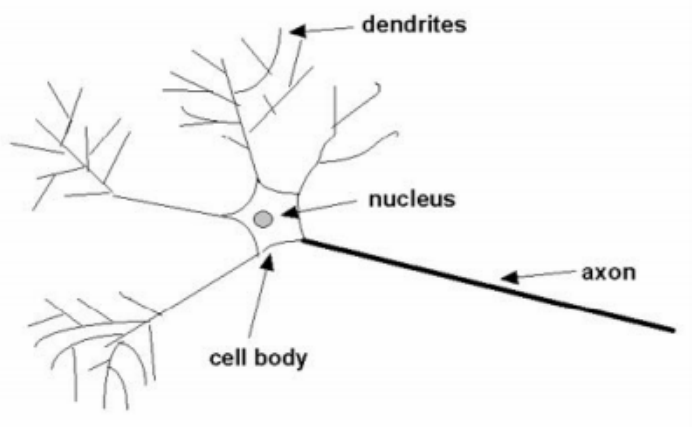


Figure 19 Biological model of a neuron. Credits Rojas

Figure 19 shows the basic model of the biological neuron: it is composed of three main elements: dendrites, cell body and axon. Dendrites collect signals from the nearby neurons and send their signals to the body of the cell. If the sum of the received signals is greater than a threshold value, the neuron produces a signal that is transmitted along the axon to the next neuron.

The neuron, seen as the fundamental unit of ANNs, is modelled taking inspiration by the biological neuron, but is characterized by simplification that make the unit more efficient from a computational point of view.

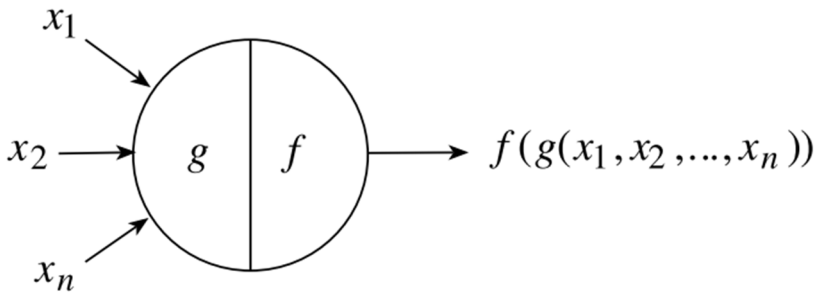


Figure 20 The artificial model of a neuron, seen as a computing element. Credits Rojas

Figure 20 shows the structure of an artificial neuron with n inputs. Each input channel i can transmit a real value x_i. A *primitive function f* computed in the body of the abstract neuron can be selected arbitrarily. Usually the input channels are associated to a weight, that means that the incoming information is multiplied by a weight that somehow defines how "important" that information is compared to the others. The collected signals are integrated at the neuron and f is evaluated. ANNs are in this sense a network of primitive functions, even if different models of ANNs differ mainly in the assumptions about the functions used, the pattern of connection, and the information transmission timing. The aggregating function g is usually the addition.

5.6.2 Network architectures

Several types of ANN exist, and a list is presented here:

Feedforward The earliest appearance of ANN, and the network with the most basic behaviour: the information moves only in the forward direction, from the input nodes, through the hidden nodes, to the output ones. There are no cycles or loops in the network.

Convolutional networks are a subset of Feedforward networks in which the connectivity pattern between neurons is inspired by the organization of visual cortex in animals, where neurons are placed in a way that responds to the different overlapping regions that compose the visual field.

Recurrent Differently from FFNs, Recurrent Neural Networks (RNNs) are characterized by bi-directional flow of information. Connections are directed from input layers to output layers, but reverse-direction connections are also present. RNNs can be used as general sequence processors.

Dynamic These types of networks include time-dependent behaviour such as transient phenomena and delay effects. The network exhibits memory, that is its response depends not only on the present input, but also on the history of the input sequence. System identification can be performed with this type of ANN.

Radial Bases Functions are a type of ANNs that uses Radial Basis Functions (RBFs, a function that has a distance criterion with respect to centre reference) as activation functions. The basic idea behind RBF networks is that a predicted target value of an item is likely to be similar to other items that have close values of the predictor variables.

Modular Biological studies have shown that the human brain is characterized not by a single, huge, network, but as a collection of small networks, in which several small networks cooperate or compete to solve problems.

Cascading networks have the peculiarity of modifying their architecture during training, starting with a minimal network and adding new hidden units one by one, as training progresses. Once a new hidden unit has been added to the network, its input-side weights are frozen. This unit then becomes a permanent feature-detector in the network, available for producing outputs or for creating other, more complex feature detectors.

Neuro-fuzzy is a combination of ANN and Fuzzy Inference System (FIS). Embedding a FIS in a general structure of an ANN has the benefit of using available ANN training methods to find the parameters of a fuzzy system.

5.6.3 Network training

One of the peculiar characteristics of ANNs is that they can be trained to mimic model behaviours: the weights that multiply each input signal will be updated until the output from the neuron is similar to the model used as a reference during the training. Generally speaking, training is an adaptive algorithm that is used to match the output of an ANN to a reference model. The algorithm iteratively compares the output of the network to the model, and by applying a corrective action on the network weights and biases, the output is adapted to match the desired one. The training is generally based on previous experience, although methods that modify the parameters of the network exist. Three types of learning algorithms have been developed.

Supervised learning denotes a method in which some input vectors (training data, that are composed by input object and matched desired output) are collected and presented to the network. The output computed by the network is observed and the deviation from the expected output is measured. Weights are corrected according to the magnitude of the error, depending on the learning algorithm. In general, every time a network is trained, different solutions can be obtained, due to different initial weight and bias values, different initialization, and different separation of the input data in the training, validation and test datasets. It will be the type of training used in the case studies presented in the thesis.

Unsupervised learning is a type of learning that does not foresee a reference to evaluate the quality of the training step by step. Since the examples fed to the network are unlabelled, the training obtained might differ from a human-based interpretation of the problem.

Reinforcement learning is conceptually similar to the supervised learning, with the only difference that input/output pairs are not presented to the network, but instead a reward (or penalty) is obtained with respect to the actions taken. Typically, reinforcement learning is a technique useful in solving control optimisation problems, that is the problem of recognizing the best action in every state of the system, optimizing some objective function.

Among the different training algorithms available, the three most common ones (that are also those available on Matlab®) are:

Levenberg-Marquardt back-propagation It is a network training function that updates weight and bias values. Like the quasi-Newton methods, the Levenberg-Marquardt algorithm was designed to approach second-order training speed without having to compute the Hessian matrix, which is approximated [59]. The Levenberg-Marquardt algorithm is very simple but robust

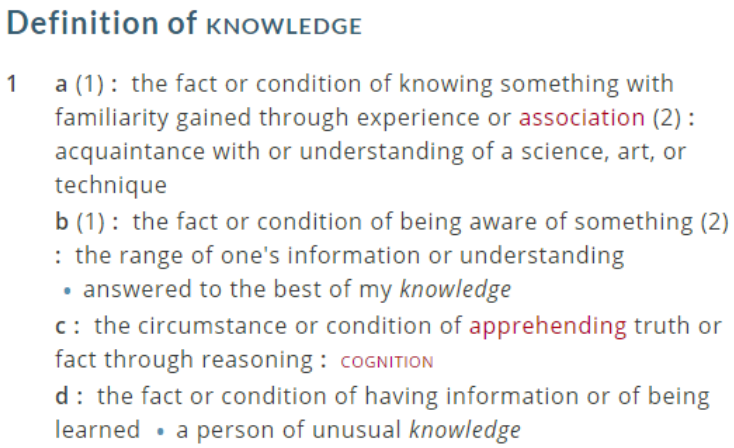
Bayesian regularisation back-propagation It is a network training function that updates the weight and bias values according to Levenberg-Marquardt optimisation. It minimises a combination of squared errors and weights, and then determines the correct combination so as to produce a network that generalises well. The function can train any network as long as its weight, net input, and transfer functions have derivative functions. It also modifies the linear combination so that at the end of training the resulting network has good generalisation qualities [60].

Scaled conjugate gradient back-propagation It is a network training function that updates weight and bias values according to the scaled conjugate gradient method. The function can train any network as long as its weight, net input, and transfer functions have derivative functions. Back-propagation is used to calculate derivatives of performance with respect to the weight and bias variables. The scaled conjugate gradient algorithm is based on conjugate directions, but this algorithm does not perform a line search at each iteration [61]. It is faster with less memory employed than previously methods. It is the training algorithm employed in the case studies presented in the thesis.

Given a specific training algorithm, two approaches exist that regulate how the data is fed to the training algorithm: in *offline training* (or *batch*), the complete dataset is fed to the training algorithm; in *online training*, the training algorithm updates the weights and biases of the network every time a new sample is fed to the training algorithm. Typically, *online training* is characterized by a slower convergence speed, also because of likely timings limitations in acquiring new samples. On the other hand, they are particularly useful when the memory available on the application does not allow to store complete datasets, but instead each sample used in the training must be forgotten before a new sample can be obtained and used.

5.7 Knowledge-based Engineering and Expert Systems

The section deals with knowledge-related algorithms and applications. The term *knowledge* itself denotes familiarity, awareness, understanding of a process or a situation, such as facts, information, descriptions or skills, which are acquired through experience or education, by perceiving, discovering, or learning [62].


Definition of KNOWLEDGE

1 a (1) : the fact or condition of knowing something with familiarity gained through experience or association (2) : acquaintance with or understanding of a science, art, or technique
b (1) : the fact or condition of being aware of something (2) : the range of one's information or understanding
• answered to the best of my *knowledge*
c : the circumstance or condition of apprehending truth or fact through reasoning : COGNITION
d : the fact or condition of having information or of being learned • a person of unusual *knowledge*

Figure 21 Definition of "knowledge" by Merriam-Webster English dictionary

Knowledge can be considered a “tangible” concept, in the light of designing space missions and space mission applications, especially when developing an application that makes use of a translation to machine code of the knowledge that a domain expert would use to solve the application underlying problem.

This is the case of Knowledge-based Engineering (KBE) and specifically of Knowledge-based Systems (KBS). This term denotes a design approach and

philosophy, and a corresponding type of systems, that use expert knowledge as a fundamental pillar. The main difference between a KBS and a conventional application can be found in their structure, where the two roles of domain knowledge and general application software are distinctly separated. On conventional programs instead, the two layers are often joined and no distinction can be easily observed in the code structure. The main consequence of this distinction is that the knowledge base, that collects all the rules and concepts that define the behaviour of the application, can be updated by domain experts without having to coding into details the program structure, because the programming expertise required for knowledge updating is consistently smaller [63].

5.7.1 Knowledge Based Systems

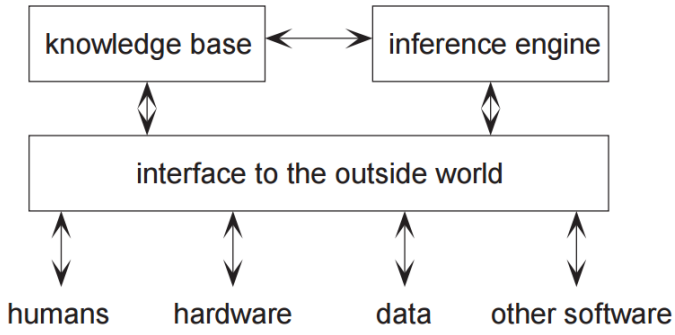


Figure 22 Basic Knowledge Based System architecture

A typical architecture for a KBS is shown in Figure 22, and will be described in the following sections. Generically speaking, the Knowledge Base (KB) is responsible of storing the knowledge in the system, and the Inference Engine (IE) is responsible of defining how to apply the knowledge.

5.7.1.1 Knowledge Base

The Knowledge Base is the portion of the algorithm that has the purpose of storing the diverse forms of knowledge: rules and facts are examples. Rules might be complex, and facts can be organized in complex structures that include attributes and relationship between entities.

An example of rule, very common and probably the simplest one, is the so-called *production rule*:

if <*condition*> *then* <*conclusion*>

One of the advantages of storing knowledge in the form of production rules is that they can often be expressed in a form that closely resembles natural language, as opposed to computer language. Facts, in a similar approach, are also stored in the KB. They can be categorized into different types: static facts, made available and fixed in time; transient facts, made available when the system is executing; derived facts, that are generated as a result of applying a rule. With respect to traditional programming, storing hundreds or thousands of facts and rules into a KBS is easier: rules and fact are represented explicitly and can be changed at will. The paradigm changes completely when the knowledge handled by the system is characterized by some degree of uncertainty. Several types of uncertainties exist:

- Uncertain evidence
- Uncertain connection between evidences and conclusions
- Value values

5.7.1.2 Inference Engine

Inference Engines vary greatly according to the type and complexity of the knowledge they deal with. Two types of inference engines can be identified:

- Forward Chaining, or data-driven
- Backward chaining, or goal-driven

A KBS that employs the data-driven mode uses the available information (the facts) and generates as many derived facts as it can. Outcomes of this process can be either satisfying or not, as the output is often unpredictable and the system might generate innovative solutions to a problem or wasting time generating irrelevant information. A typical usage of the data-driven is for problems of interpretation where data must be analysed. A goal-driven system, on the other hand, is appropriate when a more focused solution is required. The process employed by a goal-driven IE is to start from the given goal and trying to trace the information back to the current status of the application (therefore generating the plan), or assessing that no possible path is obtainable from the given goal back to the current status.

5.7.2 Expert Systems

Expert Systems (ES) are a type of KBS designed to manage and use expertise in a particular, specialized, domain. An ES is intended to act as a human expert who can be consulted on a range of problems that fall within his or her domain of expertise.

Typically, the user of an ES will enter into a dialogue in which he or she describes the problem (such as the symptoms of a fault) and the ES offers advice, suggestions or recommendation. In other applications, the ES is directly configured by the expert to act automatically, replacing the expert in taking actions driven by the stored knowledge. Additionally, depending on the application, the ES might be required to justify the current line of actions: an Explanation Module is often added to the ES to help with this purpose.

When an ES is programmed but no knowledge is stored, it is called Expert System Shell: in principle, it should be feasible to develop an ES shell, build up a KB, effectively obtaining an ES. However, all domains are different, and it is difficult to build a shell that adequately handles the various applications. Generally speaking, ES shells are not suited for embedded applications.

5.7.3 Fuzzy Logics

Fuzzy Logics address a specific source of uncertainty: the vagueness of the information. Developed in 1975 by Zadeh [64], builds on his theory of fuzzy sets developed in 1965 [65], with the objective of performing computation with linguistic variables and values, that are not unambiguously correlated to specific values. The result is that, by using the Fuzzy Logic theory, systems can be designed to operate basing themselves on values such as “big”, “small”, “enough” and so on.

5.7.3.1 Crisp and Fuzzy Sets

Fuzzy Sets carry a very distinct meaning with respect Crisp Sets. An example of Crisp Set would be a variable that qualitatively measures a temperature value on a spacecraft component. If a hypothetic control logic is set to three different actions depending on the temperature being defined as *high*, *medium* or *low*, a Crisp Set would be defined in the following way:

- $T > 50°C$ is *high*
- $10°C < T < 50°C$ is *medium*
- $T < 10°C$ is *low*

Each boundary is considered strict: if a temperature is *high*, then it cannot be nor *medium* nor *low*. In this example, a *high* temperature might trigger a completely different control action with respect to a *medium* one, and no distinction might be implemented between a temperature of 51°C or one of 150°C, as they would both be considered *high*.

Fuzzy Sets are a mean of reducing how strict these boundaries are. The theory of Fuzzy Sets expresses imprecisions quantitatively by introducing characteristic membership functions that can assume values between 0 and 1 corresponding to degrees of membership of a variable value to a condition, from being "not a member" to a "full member". The degree of membership is sometimes called the possibility that a certain value is described by the membership function. The key differences between a Crisp and a Fuzzy set are:

- An element has a degree of membership in the range [0,1]
- Membership to one Fuzzy Set does not preclude membership to another

In the temperature example, the fuzzy theory terminology is configured as follows:

- Fuzzy statement: "temperature is low"
- Fuzzy set: low temperatures
- Fuzzy variable: temperature
- Membership function: how the degree of membership to the fuzzy set is evolving with the measured temperature

5.7.3.2 Fuzzification

To recall the earlier example on temperature, a temperature of 150°C could be considered 0.99 *high*, and 0.01 *medium*, while a temperature of 51°C could be considered 0.30 *high* and 0.70 *medium*. The process of deriving these possibility values for a given value of the variable is called fuzzification.

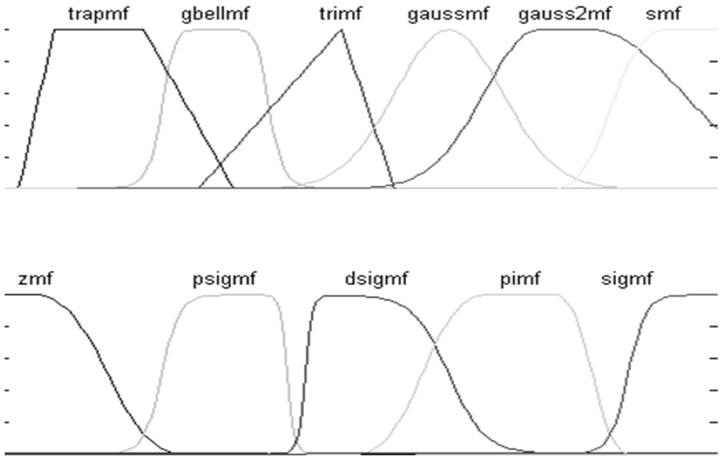


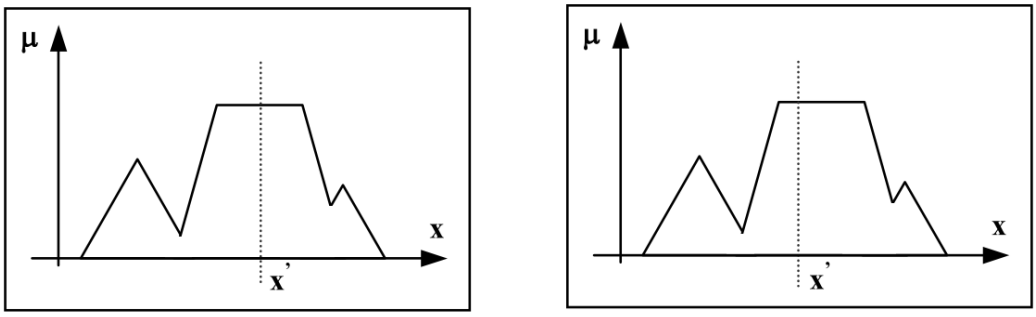
Figure 23 Examples of membership functions. Credits MathWorks

Examples of membership functions are shown in Figure 23: they can assume different shapes, and the most suitable shape of the membership functions and the number of the fuzzy sets depend on the particular application.

5.7.3.3 Defuzzification

When designing an application that employs a Fuzzy Logic -based algorithm, after defining the input variables and their membership functions, it is necessary to continue the design process downstream to the output of the application. When a control action or a decision is computed using Fuzzy Logic, the value of the action will still be expressed in fuzzified values. In order to compute back a crisp, clear value, the next process to perform is the defuzzification. Defuzzification takes place in two steps:

- Adjusting the fuzzy sets in accordance with the calculated possibilities. Several rules exist to process the various membership functions (Larsen's product operation rule is one, in which membership functions are multiplied by their respective possibility values [66]). The effect is to compress the fuzzy sets so that the peaks equal the calculated possibility values. Alternative approaches are also present.
- Using methods to correlate the fuzzified values to a crisp value. Methods applicable are Mean of Maximum (MOM) method, Centre of Gravity (COG) Method, Height Method (HM) or Lookup Table [67] (Figure 24). Other common method of defuzzification is Sugeno method [68].



(a) MOM method example (b) COG method example

Figure 24 Example of MOM and COG methods for defuzzification

5.8 Evolutionary Algorithms

Evolutionary Algorithms (EA) are a category of Artificial Intelligence used to perform metaheuristic optimization. Metaheuristic can be defined as *"a common but unfortunate name for any stochastic optimization algorithm intended to be the last resort before giving up and using random or brute-force search. Such algorithms are used for problems where you don't know how to find a good solution, but if shown a candidate solution, you can give it a grade"* [69]. The main inspiration for EA comes from genetics and natural selection [70] and at least four categories can be identified that belong to EA:

Genetic Algorithms (GA) by far the most diffused category of EA. Algorithm that treats the solution of a problem as individuals, and obtains optimal solution by applying operators such as recombination and mutation.

Evolutionary Strategies (ES) are a type of algorithm that reaches optimal solutions by applying mutation and selection operators [71], and can be successfully employed even with populations numbers as low as two individuals. The selection of individuals is performed only on fitness rankings and not on the actual fitness values.

Evolutionary Programming (EP) is another common EA [72], and is based on defining a program whose numerical parameters are subject to evolution. It is harder and harder to distinguish from ES.

Genetic Programming (GP) is an optimization method that treats the solution to a particular problem as computer programs, traditionally represented in memory as tree structures. At each node of the tree there is an operand that executes mathematical expressions.

In general, EA perform well approximating solutions to all types of problems because they are not tailored to assumptions about the function shape to be explored.

As with Machine Learning, the number of algorithms gravitating into the domain of Evolutionary Algorithms is enormous, with hundreds of algorithms and even more variations (Figure 25).

Figure 25 Non-comprehensive map of Evolutionary Algorithms and their variants

The case study presented in this thesis, that shows an application of EA, will implement Genetic Algorithms, since they are the most common type of EA and the methodology fits well in the design problem.

5.8.1 Genetic Algorithms

Genetic Algorithms are the most used and known type of Evolutionary Algorithm, to the point that the whole category is sometimes confused with GA. They owe their diffusion to the numerous field of application they've found: parameters optimization, financial prediction, scheduling, telecommunication, computer drawing, datamining, bioinformatics and so on.

GAs are powerful search algorithms: they explore the solution space quickly in search of optimal solutions [73]. GAs encode the decision variables (or input parameters) of the problem into an array that represent a full solution [74]. Each array assumes the characteristics of a *chromosome* and represents an *individual* solution among the population. The position in the chromosome of each gene is called *locus*, while its value is called *allele*. There are two encoding classes: *genotype* and *phenotype*. Genotype denotes the ensemble of all the genes of an individual, while the phenotype denotes the group of all the visible features and characteristics of the individual. A *fitness function* is the "grading system" that is

used to evaluate the fitness of an individual in the problem considered. Unlike other optimization techniques, the fitness function of GAs may be defined in mathematical terms, or as a complex computer simulation, or even in terms of subjective human evaluation. Operators are used to regulate the evolution of the population. The three genetic operators commonly used are: *selection*, *crossover*, *mutation* [75].

Selection operator is used to generate a parent population which favours good individuals. There is a *selection* pressure that rules the selection schemes: it's defined as the ratio between the probability of selection of the best individual to that of an average individual. There are two basic types of selection schemes: *proportionate* and *ordinal* methods [76]. Proportionate selection choses individuals by comparing the fitness values, while ordinal selection selects the individual by comparing the order in which they appear when the population is ranked. Several methods exist to perform the selection, and the *tournament* is one of the most common: two (or more) individuals are randomly chosen and compared to each other; the best is placed into the parent population. Other selection methods are *roulette wheel* (each individual has a chance of being selected proportional to its fitness), *stochastic universal sampling* (the probability is proportional to the fitness, but an equally space pointer is used).

Crossover operator is used to generate offspring from parents, and it can operate in different way depending on which type of strategy is chosen: *single point* crossover selects a locus on both parents' chromosomes and swaps the strings after the locus; *two-point* is similar to the former, but selects two loci in the strings and swaps only the middle portion; *uniform* and *half uniform* uses a fixed mixing ratio between two parents: unlinked single- and two-point crossover, the uniform crossover enables the parent chromosomes to contribute at the gene level instead than the segment level; *three parent* uses three randomly chosen parents and generates offspring by comparing a gene in two parents, and selecting the gene in the same locus from the third parent if the first two are different from each other.

Mutation operator is used to alter an individual, by changing the value of one or more (but a limited number with respect to the total gene number) optimization variable in a random way. It is typically applied with a low probability (up to 5%) and it does not have a great influence on the performance. It is useful to avoid the issue of having the population stuck on a local minimum.

5.8.2 Design Suggestions and Improvements

GAs behaviour and optimization performance can be improved by implementing variations in the traditional functioning of the algorithm. *Elitism* can be added, to preserve the best individuals of a population and to replace the worst individuals of the following generation to preserve a good solution; *extended random initialization*: performing several random initializations until a significant solution is found or a maximum number of tries is reached, to assure the presence of good individuals in the initial population (a portion of the population can be still initialized randomly only once, to preserve diversity); *mass mutation*, to ensure that the diversity of the population stays high, by discarding most of the old individuals and replacing them with random new ones.

There are additional things to consider when dealing with GA design: practical suggestions that help to avoid common mistakes. Designing appropriate encoding schemes is useful: representation by binary codes, real-values and program code are available, and the length of an individual can be constant or change. Experience suggests to prefer identical genotypes and phenotypes, and fixed length individuals. Critical attention must be placed in designing the fitness function. Tournament selection is perceived as one of the most effective selection methods. Tournament selection with a tournament size of 2 individuals is advisable. Building-blocks crossover (that does not interrupt a good inter-gene linkage) is especially advised if the evaluation of the fitness function is computationally expensive, otherwise a one- or two-point crossover is a good approach, provided that the crossover probability is set to be relatively high. When crossover or mutation operators generate infeasible solution (because of constraints) the two approaches are to apply a penalty or to repair them: repairing the individuals is advisable unless the development of the function is difficult. Finally, the application of population-sizing models is suggested.

Chapter 6

Case Study: Event Detection with Neural Networks

Neural Networks can be used for different types of applications, and each category of NN excel in one or more specific domain. This chapter focuses on performing Event Detection (ED) during the mission: the applications presented here refer to detection of external mission events. Event detection, in particular external events related to payload observation, is a fundamental characteristic of highly autonomous spacecraft.

6.1 Background

In the previous chapter of this thesis, several applications of Artificial Intelligence to increase the mission autonomy have been introduced, and it was shown how enhanced autonomy could specifically benefit the nano- and small-satellite missions. Making the spacecraft autonomous is a topic of paramount importance especially for missions beyond LEO, where the current limited autonomy capabilities are a severe stopper for the diffusion of nanosatellite interplanetary missions. Examples of these missions are those targeted to the Moon, Near Earth Asteroids (NEAs), Mars and Jupiter, with his satellite Europa. These destinations have already been chosen by space agencies as ideal candidates for CubeSats and nanosatellites missions that will reach them in the near future as a secondary payload of traditional flagship missions. Moreover, CubeSat missions have been proposed and are being developed by NASA and ESA as part of their space

exploration programs. A perfect example of this commitment is the deployment of thirteen spacecraft by the Orion vehicle during the Exploration Mission 1 (EM-1), scheduled for launch in 2018 to the Moon by NASA [77]. The deployment orbit of the CubeSats will be a lunar transfer orbit from which they will start their own independent missions, most of which have either scientific or technological objectives. ESA has already started studies to deploy CubeSats in the vicinity of the Didymos binary asteroid as secondary payload of the Asteroid Impact Mission (AIM) spacecraft within the NASA/ESA Asteroid Impact & Deflection Assessment (AIDA) mission. The CubeSats, named COPINS (CubeSat Opportunity Payload Intersatellite Network Sensors), will pursue technological and scientific objectives and will use the Aim spacecraft as a relay to send housekeeping and payload data to Earth. Despite this mission being cancelled by ESA at the end of 2016 [78], another interplanetary CubeSat mission is being developed on the traces left by the AIM/COPINS study: M-ARGO, Miniaturised Asteroid Remote Geophysical Observer, a stand-alone deep space CubeSat system for low cost science and exploration missions [79]. Objective of this mission is a rendezvous with a NEO for physical characterization and resource assessment. Another example is the first CubeSat mission to Mars, MarCO, that will be part of the InSight mission of NASA/JPL. MarCO aims at testing telecommunications capabilities from deep space [6]. These nanosatellites will face new challenges with respect to current LEO missions, such as surviving in deep space environment, communicating to Earth from longer distances, and using their own propulsion systems. In addition, a paradigm shift is required when taking into account the operation design, as most GCS are not adequate to receive signals from space from a multitude of spacecraft, and no line-of-sight periods will occur depending on the specific mission.

Taking into account the emerging needs of new nanosatellites missions, the application presented in this case study aims at supporting the development of more autonomous spacecraft, able to decide and execute tasks independently from ground control and from mothership authority. The presented algorithm forms the foundation for event-based autonomous operations. This case study presents an activity whose objective was to design an algorithm to enable spacecraft with event detection capabilities, with the intent of performing autonomous mission operations. The key reference mission is AIM/COPINS and the event to be detected is the impact event on the asteroid. In particular, two different applications are here presented:

- Detecting the impact event, that is the change in surface characteristics of the observed area

- Detecting a plume event, identifying the direction towards which the plume is expanding

The problem behind this application is in practice an image change detection problem, that can be treated in several ways. Many image/feature recognition algorithms exist, and they are becoming more and more useful in various applications, both in the industrial and in the scientific field. A distinction can be made between algorithms that use an *a priori* knowledge of the features to be identified, and those that use statistical or other methods to perform the detection, i.e. without initial training. Among the former category a few examples of applications are: convolutional neural networks for crater detection [80], and faces identification [81]; random forest classifiers for space image processing [82] [83]; adaptive background subtraction for video surveillance [84]. Among the latter, interesting examples are: visual salience maps to model visual attention [85] [21], unsupervised neural networks for fault diagnosis [86], pixel comparison for medical diagnosis [49] and change detection in overhead images [87].

The use of Artificial Intelligence algorithms is usually computationally expensive: the aforementioned examples involve complex operations or use datasets containing thousands of samples. For this reason, exploiting AI capabilities on-board a small spacecraft, where the computational power is low, requires the implementation of algorithms that are specifically designed to have a limited impact on the processing resources [88]. The presented application is specifically developed with this precise objective: avoiding the computational resources overhead due to the huge size of datasets commonly used in classification problems with NN, by developing a custom-designed network and innovative training approach.

6.2 Reference Missions

For the intended type of application, two reference missions were considered: a mission that involves an interplanetary CubeSat that performs observations on an asteroid on which there will be an impact event, and an interplanetary mission to a comet, on which events such as plumes and gas ejections can happen.

6.2.1 Impact Mission

One of the reference mission for this research was given by COPINS, which was a secondary payload of the ESA AIM mission. AIM was one of the two

spacecraft of the AIDA joint effort of ESA and NASA, aiming to perform an asteroid characterization and redirection experiment. ESA was providing the monitoring/observing spacecraft (AIM), while NASA was supposed to launch the impactor probe (DART) that would collide with the secondary body of the system [89]. The COPINS mission consists of multiple CubeSats (up to two 3U platforms) carried to the asteroid by AIM, which will deploy the nanosatellites at 10 km from the secondary body surface, up to one month before the impact of DART. The objectives of the CubeSat mission are to provide scientific support to the AIM primary mission, either by repeating one or more of the main spacecraft's measurements, by performing additional science measurements, or by recording and taking pictures of the impact event. In addition, the CubeSats will also perform technological demonstrations, such as satellite interlink communication. The communications of the COPINS with Earth are relayed through the AIM spacecraft. The architecture of this mission is definitely complex, as numerous challenging elements are included in the scenario: four or more satellites joint operations, inter-satellite links, limited data rates, and peculiar environment (for example, low and irregular gravity field, which makes the orbit control critical). Given the complexity of the mission architecture and concept of operations, increasing the COPINS autonomy would be beneficial to the entire mission, and for this reason this mission has been chosen as a test case for the developed algorithm. For the purpose of the research, it is assumed that the COPINS's payload objective is to detect the impact of DART on Didymoon (the secondary body of the Didymos binary system, other times referred to as moonlet) and to determine the changes in the physical properties of the asteroid surface. Since the COPINS-Earth communication is characterised by the fact that the main spacecraft serves as relay, the amount of data that can be sent to Earth by COPINS and the possibility to command COPINS from Earth are both affected by the availability of AIM. The autonomous detection of the impact event would enable:

- To implement switching between operative modes. Switching between a hypothetical basic operative mode to the science operative mode could be performed with enhanced flexibility and increased reliability, without relying on commands from ground. The post-impact operations would start autonomously.
- To prioritise downlinked data. Given the limited data rate available for the downlink, the pre-selection of payload data would avoid sending meaningless information to ground in favour of data related to the completion of scientific objectives.

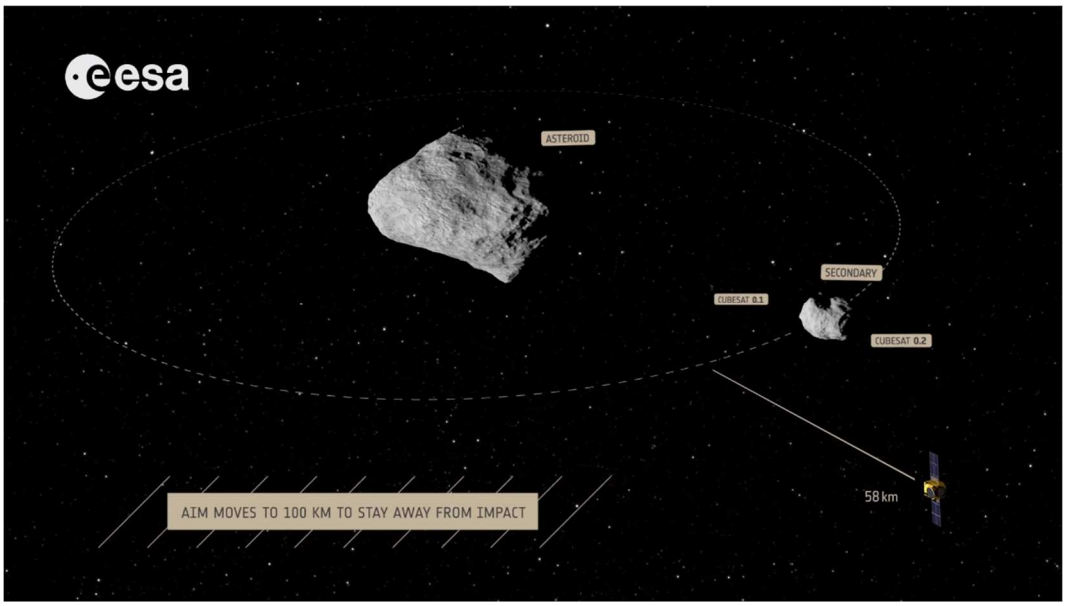


Figure 26 AIM and COPINS Design Reference Mission. Credits ESA

6.2.2 Comet Mission

Second reference mission used in this thesis is a hypothetic mission to a comet-like body of the solar system. These bodies are known to be the potential source of jets and plumes, as demonstrated in several occasions to date (Figure 27, Figure 28).

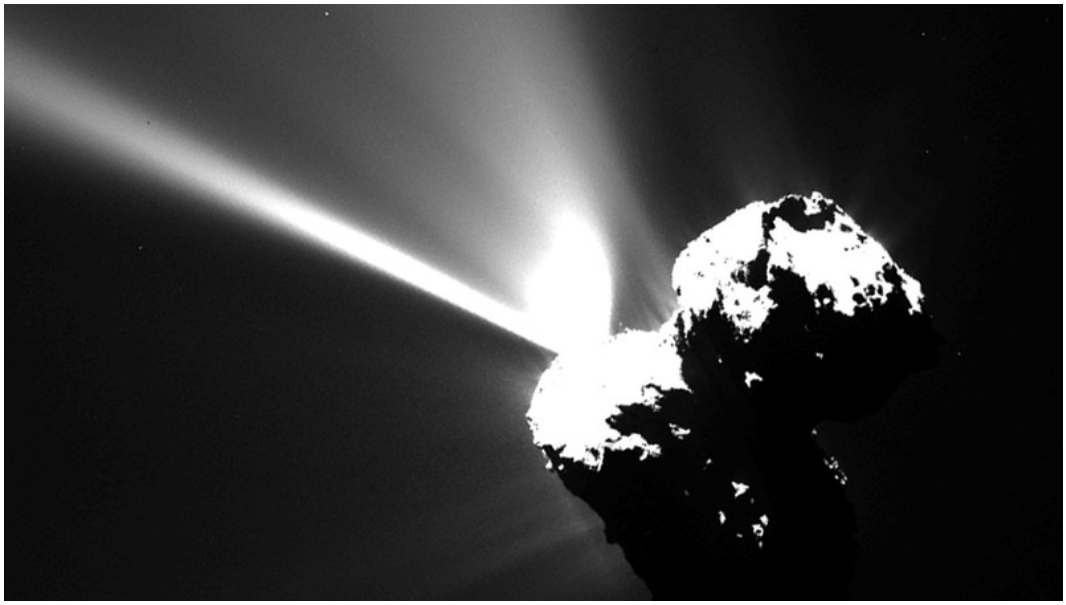


Figure 27 Jets emitted by comet 67P. Source ESA

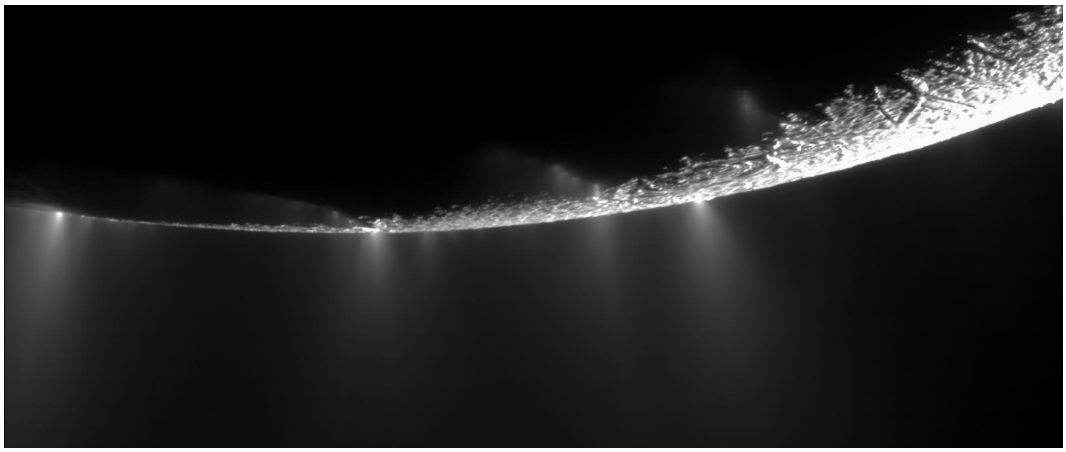


Figure 28 Plumes emitted by Enceladus, a moon of Saturn. Source ESA

The origin of these events can be of various nature, and among the known causes are solar activity [90] or man-made impacts [91]. For the purpose, also in this mission the CubeSats are considered deployed directly in situ by a mothership.

6.3 Neural Network architecture selection

When designing a network for an event detection case study, several factors must be taken into account. In the presented case study, main driver is the computational cost needed to train and run the algorithm, as it will be implemented on the embedded processor of a nanosatellite with limited resources. The three criteria considered are reported in Table 10.

Table 10 Criteria for network architecture selection

Criterion	Value
Training performance	High
Execution performance	High
Network complexity	Low

6.3.1 Impact Event detection network

For the application of ED, a simple feed-forward architecture is chosen, as shown in Figure 29. The parameters of the network are presented in Table 11.

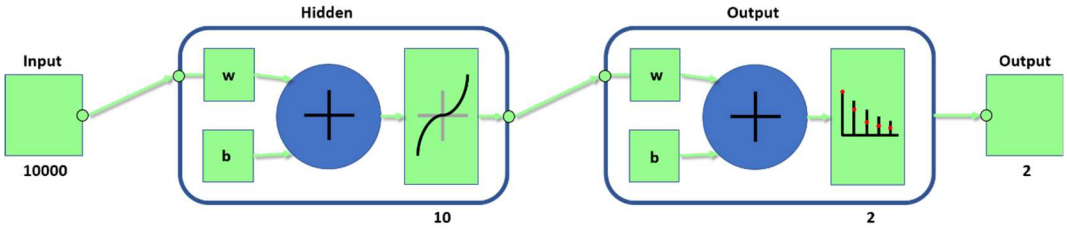


Figure 29 Feed-forward network architecture

Table 11 Network parameters

Parameter	Value
Architecture	Feed-forward (FF), one Hidden Layer (HL)
Dataset element type	Image
Dataset element dimension	100x100 pixel
Hidden layer size	10 neurons
Output layer size	2 neurons
Training algorithm	Scaled Conjugate Gradient (SCG)
Threshold function	Symmetric sigmoid

The final number of layers and neurons per layer is the result of the analysis performed over a set of possible network architectures. To select the most performing network, a statistical analysis over all the possible architectures compatible with the main requirement (compatibility with the CubeSat C&DH performances) was performed. In particular, networks with one or two hidden layers were tested, up to a maximum number of neurons of 15 for the first layer, and of 10

for the second layer. Figure 30 shows the average performance of each network cluster for a two-hidden layer architecture: each dot represents the average of architecture performance in function of number of neurons in the second layer. The average is calculated over 4500 simulations (300 simulations for number of neuron in the first layer, spanning from 1 to 15). The result of the analysis confirms that for a binary classification problem, networks with one hidden layer show the best performances on average [92]. Figure 31 illustrates performances of networks with a single hidden layer in function of the number of neurons in the layer in the form of boxplots. Boxes represent data from second and third quartile, while whiskers cover data in first and fourth quartile. Samples are considered outliers when their distance is greater than 1.5 times the interquartile range, and they are represented as dots. The red line represents the performance median. For each architecture, 300 simulations have been run. From this graph, it is possible to deduct that networks with more than 4 neurons are suitable for the final architecture, as boxes are condensed into the median line.

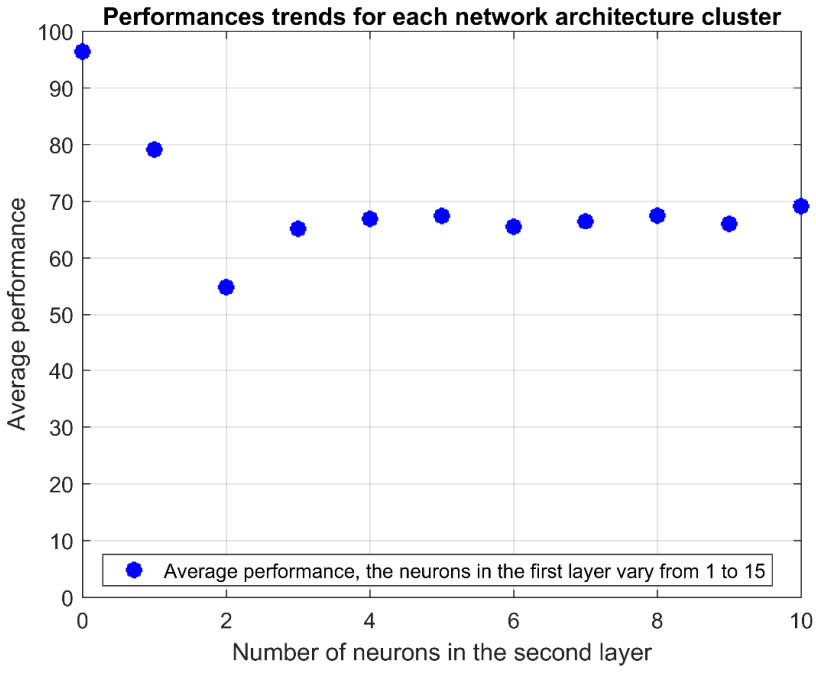


Figure 30 Performance trends for networks with two hidden layers. Each dot represents a cluster of networks with 1 to 15 neurons in the first layer, and the X-axis number of neurons in the second layer.

The number of 10 neurons for the hidden layer size was chosen as a good compromise between complexity of the network and associative memory [93]. As

the learning ability of a network increases with number of neurons, a margin was taken to consider inherent uncertainty of early mission design stage, thus selecting 10 neurons instead of 5, which is the minimum acceptable number.

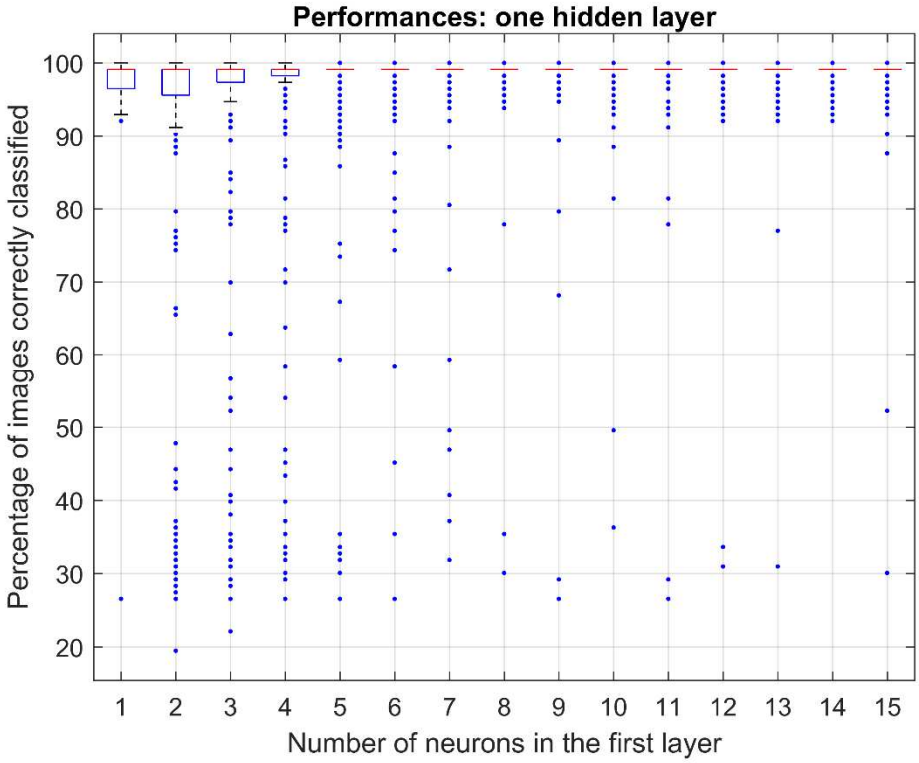


Figure 31 Average performances with respect to network architecture. Each box plot is the result of 300 network initializations. Red line represents the median, box lines represent first and third quartiles. When no box is drawn, all data except the outliers are collapsed in the median value. Outliers represent samples that lie further than 1.5 times the interquartile range.

6.3.2 Obtaining additional information from the detection

When developing an event detection application, objectives must be defined to correctly select the network to be used. In particular, as seen above, when the aim is to correctly identify just the appearance of the phenomenon, small networks are performing already interestingly. When additional information must be extracted, bigger networks must also be considered. Among the interesting information that can be obtained when detecting an impact or a plume event, is the direction towards which this jet is moving. An interesting future study can be correlating the size of the network with the resolution of the direction towards which a plume has been

expelled. For the applications considered, networks with a total number of 100 neurons where used. In general, no optimization was performed on the networks presented for this second application.

6.4 Event modelling

The asteroid impact sequence needed to be modelled in order to develop and test the ANN algorithm.

6.4.1 Asteroid impact modelling

The Didymos binary system is modelled as defined in the literature by the Didymos Reference Model [94]. The main body is represented as a fairly regular spheroid of roughly 800m in diameter, while the secondary body (of which no radar or optical images are available to date) is modelled as a bumpier, rubble-pile like body, elongated in the direction towards the main body of the system (Figure 32).

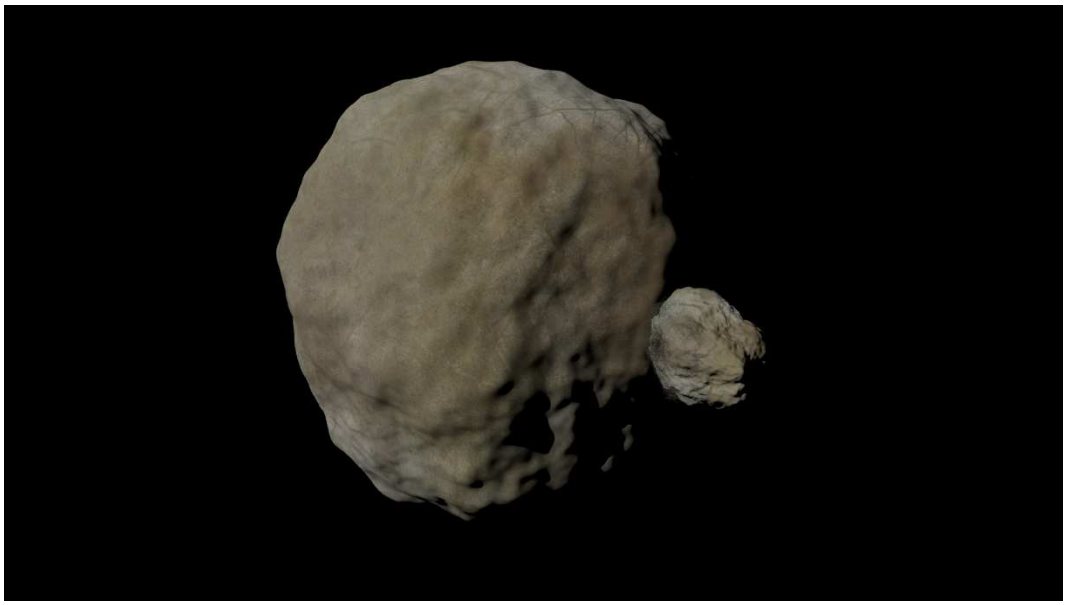


Figure 32 Asteroid modelling

The impact event is modelled according to information found in literature [89]. A spacecraft of the size of DART has been included in the simulation to collide with Didymoon at the speed of 7 km/s. All the modelling has been realized using the open software blender® (Figure 33).

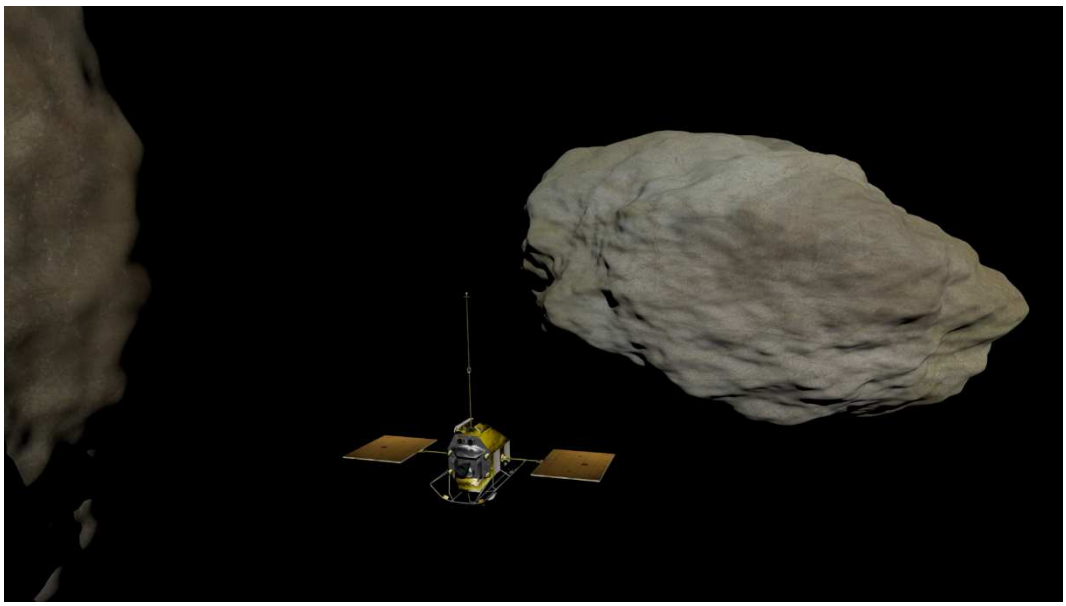


Figure 33 Impact on the secondary body

An overview of the impact location, observed from two different capturing points, is shown in Figure 34.

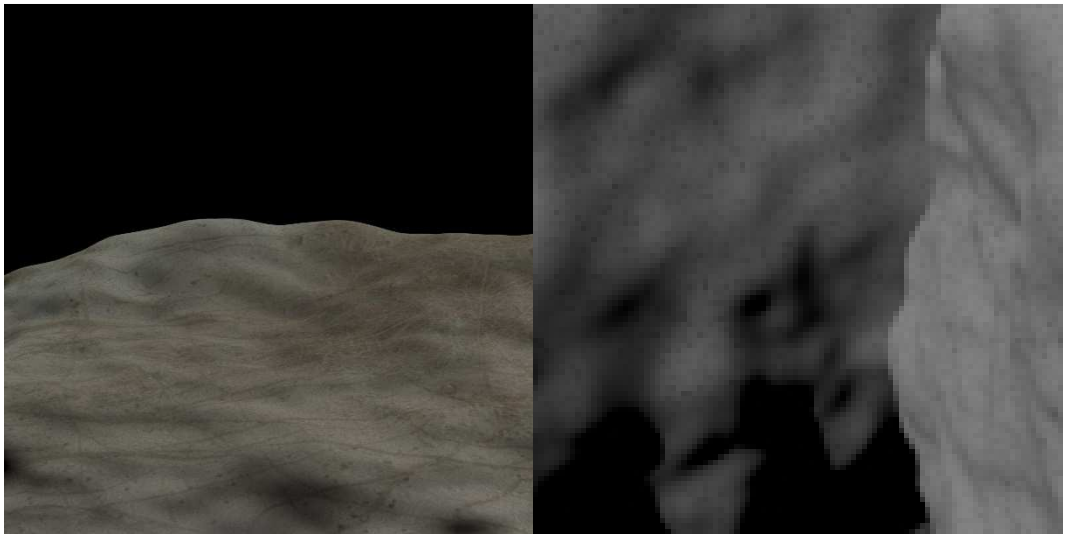


Figure 34 Impact location, as seen from two different observation points

6.4.2 Plume event modelling

The shape and the plume event have also been modelled in blender®. The characteristics of the object as matched to resemble common rubble-pile asteroids.

The asteroid is set on a slow rotation on all the three axes, and the jet is emitted from a randomly chosen location on the asteroid surface (Figure 35).

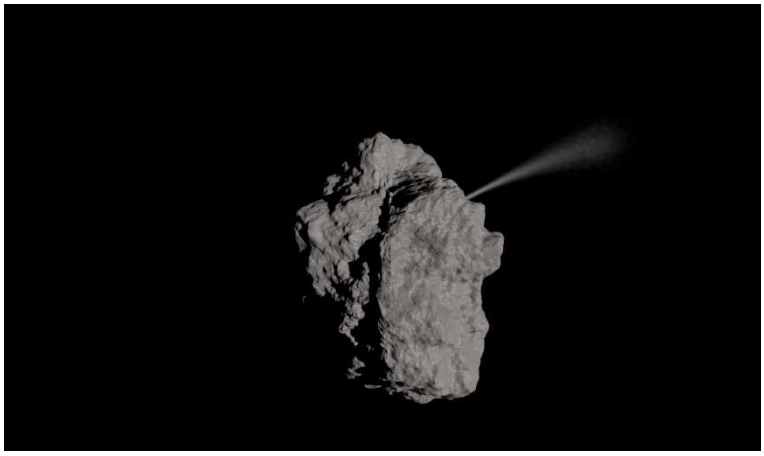


Figure 35 Asteroid modelling and plume event

To further validate the methodology, a plume event detection has been simulated on comet 67P model (Figure 36). The intent, in this second case, was to train a network for a real-life mission: for the Rosetta mission, actual images of plumes are available.

Figure 36 Plume event simulated on the comet 67P

6.5 Innovative Training Approach

One of the fundamental steps in the design of an ANN is the definition of the training strategy, which heavily affects the robustness and reliability of the network [88].

The ANN algorithm presented required the definition of a special training approach, given the peculiar application under study.

Objective of the algorithm is the identification of an event that will occur on an asteroid (or comet) at a certain time in the future. Table 12 summarises the main mission data from which requirements and constraints for the design and implementation of the ANN have been derived.

Table 12: mission inputs for the ANN definition

Event to be detected	**Impact of DART on secondary body (Didymoon) of the Didymos binary system; plume emission event on asteroid or comet**
Event detection instrument	Optical camera (the algorithm is also compatible with IR cameras)
Event Time	Within one month from deployment of COPINS at the asteroid (exact time will be unknown until in situ); No predictions are available on the next plume event
Info	Both Didymoon and the target asteroid/comet have never been observed, and no images are available nor will be before in situ

From the information summed up in Table 12, it is evident that the ANN cannot be trained on ground using actual images of the celestial bodies, as they do not exist (concerning the comet 67P model, the application is developed forcing the acquisition of the pictures in situ). Using a training dataset extrapolated from models of the asteroid would be risky, as the network would get trained on a specific shape of the asteroid that might result different from the actual shape: the possibility exists that the impact will not be identified due to incorrect training of the algorithm. Moreover, several conditions will likely be different from those simulated on ground, especially with regards to the surface features (for example areas of different composition) and light/shadowed areas (for example different crater patterns).

The proposed solution for the training task takes into account the mission scenario and concept of operations. As the spacecraft will reach its final orbit before the event to be detected, it is possible to define a sequence of manoeuvres and operations that allow the spacecraft to construct the training dataset directly in-situ, either acquiring pictures of the foreseen impact area on Didymoon, or collecting pictures of the comet prior to plume events.

Since the network employed in the algorithm performs pattern recognition, it is mandatory to differentiate between more than one class in the dataset. In particular, in order to detect an impact event with a pattern recognition algorithm, two classes of images must be used during training: images taken before the event, and images representing the event itself, in order to correctly train the network. The impact images must be artificially created in situ before the event occurs, employing an algorithm described in the next paragraph.

For a feed-forward network, considering the connections from the input to the hidden layer, they are directly mapped to the input data: in this sense, for an image, each pixel would be directly assigned to several weights. This means that, during the training to identify the event, the weights need to be raised for the pixels that would change during the event. This operation is done automatically during the training. In the proposed case studies, the only missing piece is indeed the collection of post-event images to construct a two classes dataset for the training.

6.5.1 Impact event training

For the impact event, since the coordinates of the impact on the asteroid are known, it is possible to artificially super-impose a pattern of debris-like shapes to force the weights update in particular areas of the image, as shown in Figure 37. As shown in [89] the physical properties of asteroid's surface upper layer strongly influence the characteristics of ejecta. Shape, opacity and granularity of the overlay are chosen accordingly to information found in literature to reflect the dynamics of the event to be observed. Two geometries, rectangular and truncated cone, were considered to assess the role of overlay shape in the algorithm performance.

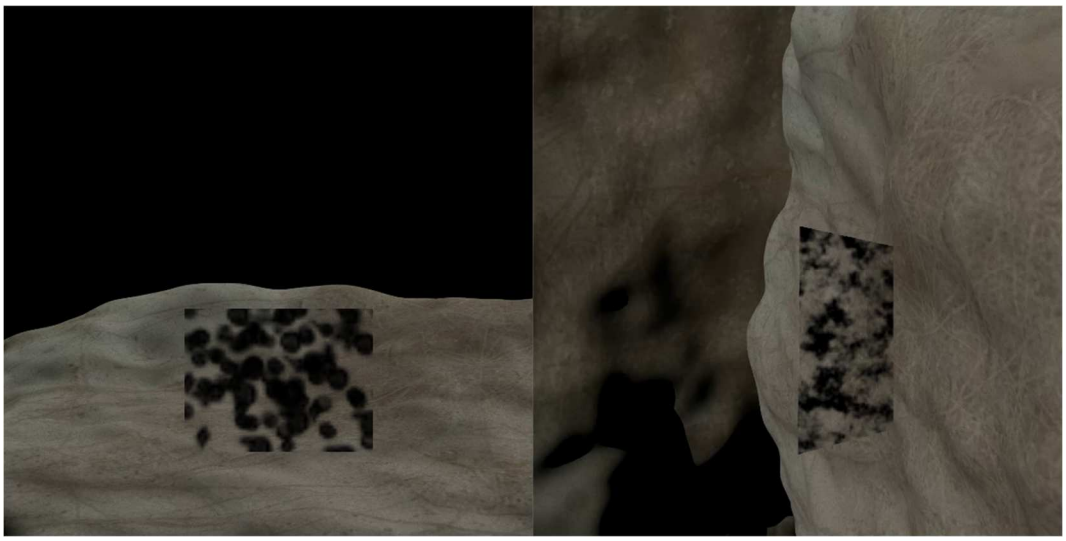


Figure 37 Directing the neuron training with pseudo-random colouring of the impact location: rectangular and truncated cone shapes

The effect is that the training algorithm will detect the differences in the modified area of the image, and the resulting weights and biases will be arranged in a way to favour the identification of changes in that particular area. This operation is effortless from the computational point of view, as the pattern can be super-imposed by using simple scripts, and it does not take into account the underlying image, resulting in a very fast operation. Figure 38 and Figure 39 demonstrate the validity of the approach, plotting the weights of the network after training. Figure 38 shows that the training assigns higher weights to the impact area, with a direct match between the overlay shape and weights. An interesting result is shown in Figure 39, where another neuron of the same network is considered. In this case, the training assigns high weights to a specific vertical zone of the camera field of view. This result shows that different neurons of a network can be trained in different ways by the training algorithm, while maintaining the desired performances.

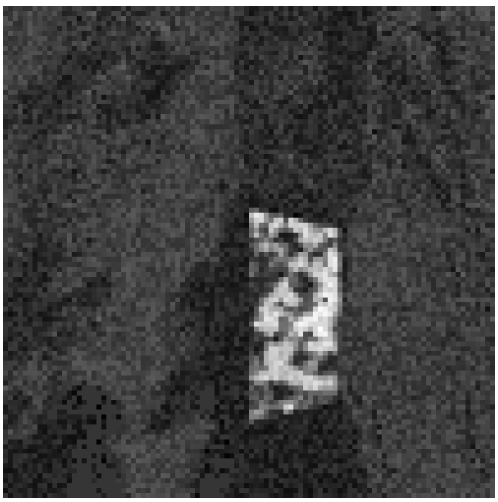


Figure 38 Trained network, input to hidden layer weights of a simple neuron. Darker pixels correspond to lower weights. Direct match between overlay and weights.

Figure 39 Trained network, input to hidden layer weights of a single neuron. darker pixels correspond to lower weights. Interesting outcome of the training.

6.5.2 Plume event training

For the plume event, since the coordinates are not known a-priori, the training approach must consider a set of probable locations. The overlay approach is performed for several directions of generation of a plume. Moreover, as the comet body is rotating in the camera frame, the generalization must be carried out both for the plume direction and for the rotational state of the body underneath (Figure 40).

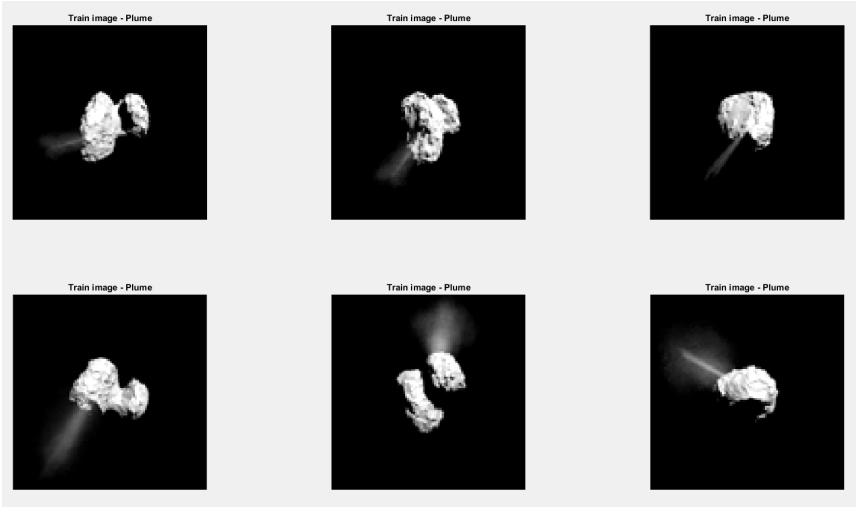


Figure 40 Examples of 67P images with an artificial plume overlay

The result of the training can be validated even before testing the performances of the network by displaying the weights final value. In this case, as for the impact event, the result clearly shows the correct training: it is interesting to notice how, given a set of single-plume images, the final weights are defined in a configuration that includes all the training images used. The result appears as a corona of high weights around the asteroid shape (Figure 41).

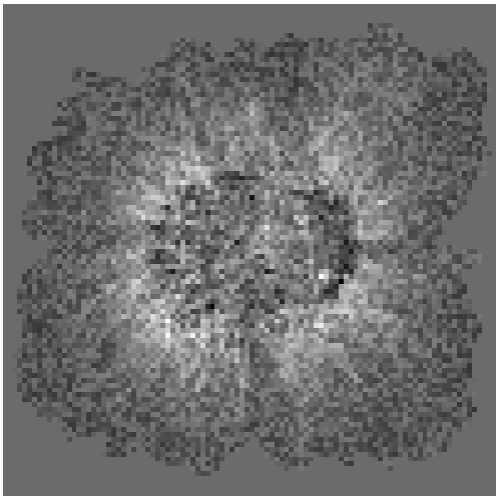


Figure 41 Trained weights for the plume detection problem. The uniform grey areas around the centre of the image are a result of having removed constant lines throughout the dataset

6.6 Results

6.6.1 Performance considerations

In order to foresee the implementation of the algorithm on an embedded processor, it is mandatory to address the performance concerns that are typically raised when considering ANN. The algorithm has been designed keeping in mind the computational cost: computational complexity of the Scaled Conjugate Gradient algorithm is evaluated at $O\left(N^2\sqrt{k}\right)$, where N is the training image matrices' rank and k is the condition number [95]. Worst case for N^2 is the image pixel total count, and for k is $O(10^4)$. The training process takes less than 5 seconds (valued considering the results of 10000 training sessions) and the resulting ANN executes in 0.02 seconds on a laptop with a core of 2.5GHz. The required RAM has been estimated in less than 1 MB. These values are compatible with the intended application of this algorithm, taking into account that state of the art processors on COTS on-board computers for nanosatellites feature 1GHz clock speed and exceed 256 MB of RAM. An estimate of the execution times on a typical nanosatellite processor is 12.5 seconds for training and 0.05 seconds to process a single image.

6.6.2 Impact Event Detection

The impact event has been simulated and tested from two capturing points (depending on the position and orientation of the observing spacecraft around the asteroid). In the first point, both bodies of the asteroid binary system are in the field of view of the satellite, with the main body in the background (Figure 42). In the second case, only the moonlet is in the field of view of the satellite, with the dark sky in the background (Figure 43). For both cases, a video of the impact has been realized, with a framerate of 25 frames per second. Frames of the post-impact evolution were then selected for the testing of the algorithm. The algorithm has been developed and tested in a Matlab/Simulink® environment, by using datasets generated via the blender® asteroid model.

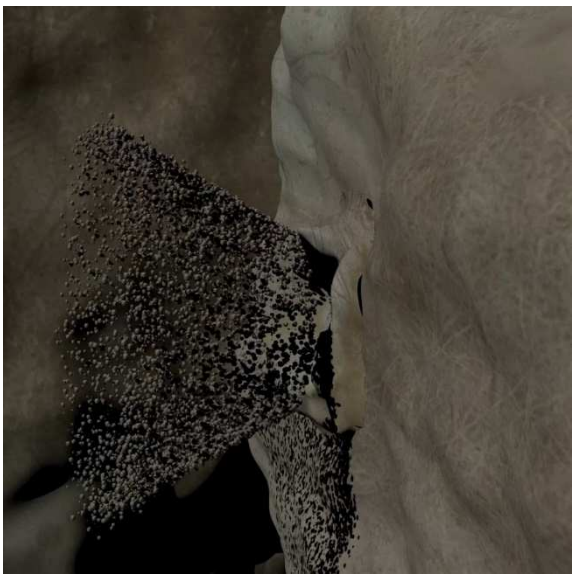


Figure 42 Impact event from first capturing point

Figure 43 Impact event from second capturing point

Four simulations have been run, changing the point of view of the impact from space and the shape of the overlay pattern representing the impact effect on the asteroid surface used in the ANN training process (Table 13).

Table 13 Mission scenarios parameters and results

Simulation #	Background	Training shape	Result
1	Body	Rectangular	Success
2	Body	Truncated cone	Success
3	Sky	Rectangular	Success
4	Sky	Truncated cone	Success

The simulations show the effectiveness of the ANN developed, as the images are correctly classified by the algorithm in the appropriate categories (Figure 44 and Figure 45).

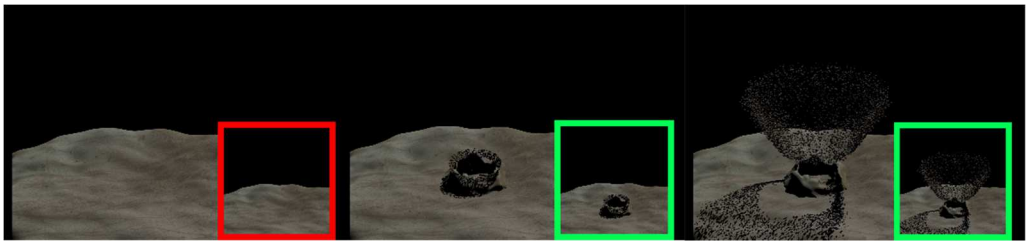


Figure 44 Impact event, dark sky in the background. Continuous line: impact detected; dashed line: no detection

Figure 45 Impact event, main body in the background. Continuous line: impact detected; dashed line: no detection

Furthermore, given the fact that the algorithm will run on board a spacecraft, it is important to test the algorithm against the disturbances due to the pointing errors that may arise during the mission. In particular, it must be guaranteed that the algorithm does not trigger false positive or fails to detect the event in case of images with different framing. The algorithm has been tested changing the orientation of

the camera on board the satellite. The range of the oscillation tested is ±12 degrees, with steps of 1 degree in the up-down pointing. To overcome the issue of oscillations affecting the detection of the impact event, the solution implemented includes images with different framing in the training dataset. In this case, the network is trained to compensate for the pointing uncertainties (Figure 46 and Figure 47).

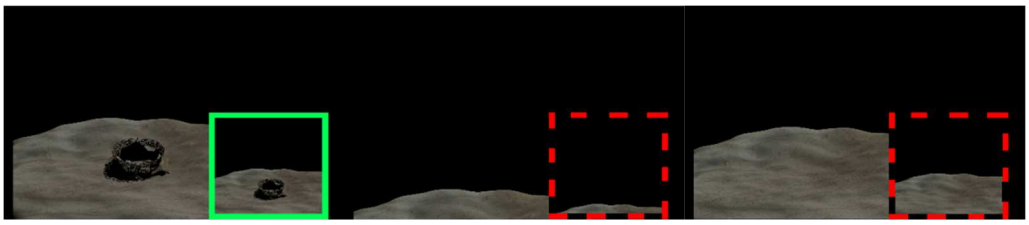


Figure 46 Robustness to imprecisions in camera pointing. Continuous line: impact detected; dashed line, no detection

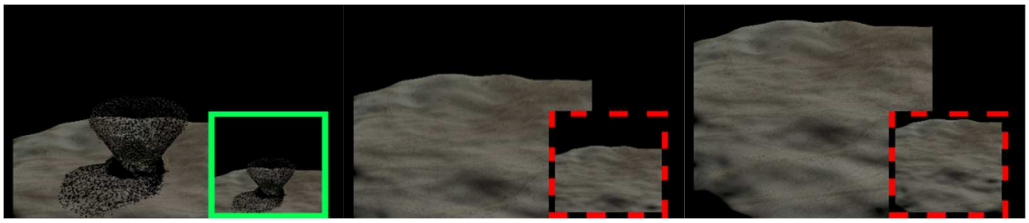


Figure 47 Robustness to imprecisions camera pointing (cont.). Continuous line: impact detected; dashed line: no detection

The algorithm obtains an average detection performance of over 98% in all the four event cases. Figure 48 depicts the confusion matrices for simulations 2 and 4 as defined in Table 13. For each matrix, the Output Class represents the decision taken by the algorithm, while the Target Class is the correct decision for each image. Class 1 represents the impact case, and Class 2 represents the non-impact case. The green quadrants represent images correctly classified. The red quadrants represent false positives and false negatives. Grey boxes show the classification performance for each class. The overall performance of the algorithm is given in the blue boxes.

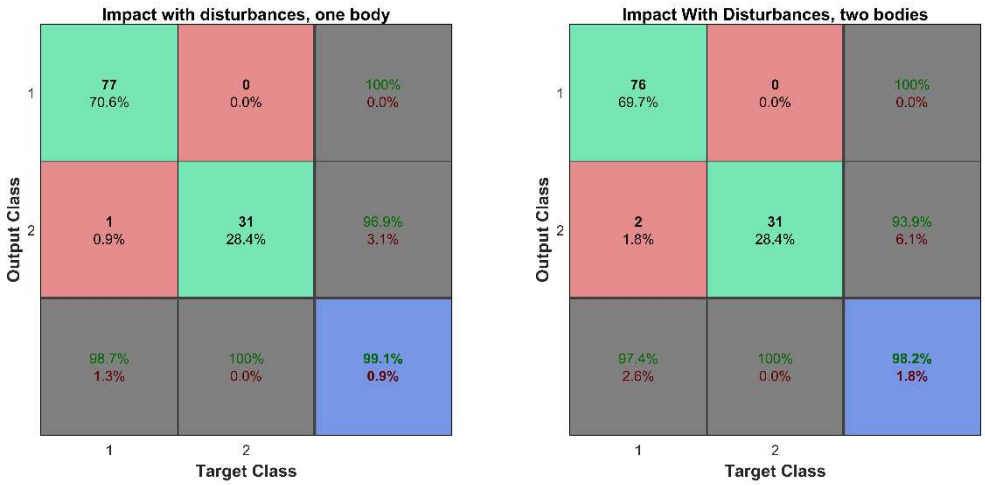


Figure 48 Confusion matrices for one body and two bodies simulations with disturbances. Class 1 represents the impact event, Class 2 represents the no-impact images

6.6.3 Plume event detection

The plume ED problem was constituted by a dataset of 1600 images used during training, divided in the following way: 98% for training, 1% for validation and 1% for testing. An additional dataset composed of 400 images was used for testing, and the ANN performance was measured on the test dataset. Figure 49 shows the confusion matrix for the 67P plume event.

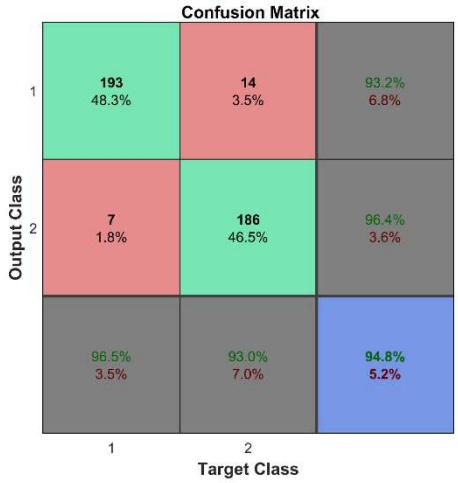


Figure 49 Confusion matrix for plume event on comet 67P

The algorithm has then been validated by evaluating its performance on real images taken by the Rosetta mission, showing plume events as experienced by the spacecraft. The detection of the events was successful, as seen in Figure 50.

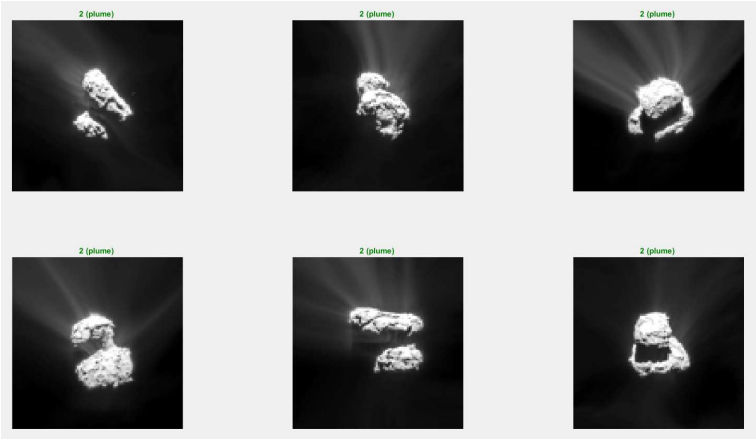


Figure 50 Detection of plume events: real images taken by the Rosetta mission

6.6.4 Review

The applications presented in this chapter provide clear examples of both the usefulness and the applicability of NN in the domain of event detection for space applications. On the other hand, the decision on which architecture is the most efficient and effective in performing different tasks needs to be object of further, deeper, investigation. Despite this, some insights can be already drawn from the research performed, and this can help towards the objective of pre-selecting NN architectures in relation with the problem to be solved. Finally, it has to be noted that the purpose of this thesis was mainly to perform feasibility analysis: for this reason, a comparison between the detection capabilities of NN and other ML algorithms needs to be performed. If the usefulness and performances of heavy architectures (such as Convolutional NN used to solve image classification problems) is well established, the research on NN for space applications, and in particular for *embedded* ones, needs to be expanded to reach a similar level of heritage.

The following table summarizes the capabilities of ANN to perform Event Detection.

Table 14 Summary of FF ANN algorithms characteristics when applied to ED.

Review Parameter	Comments
Benefits	**Training-defined Behaviour** – The behaviour of the system can be implemented to match the desired outcomes by training, and not by hard-coded programming. **Robustness** – The algorithm is, in most cases, inherently robust to disturbances, provided a correct training has been performed. The complexity of performing a correct training with respect to defining a robust algorithm in a theoretical way is reduced.
Limitations	**Architecture** – The size and complexity of each element of the dataset requires different solutions in the selection of the algorithm architecture. Event detection on an image of 100x100 pixels can be performed with small FF networks. Higher resolutions images require the use of other types of ANN (e.g. Convolutional NN).
Applicability	**Scope** – FF ANN are suitable for pattern recognition. Applications such as sensor monitoring (where a time-dependent behaviour is present) are better solved with other ANN architectures (e.g. NARX).

Chapter 7

Case Study: Failure Detection with Expert Systems

7.1 Background

The topic of failure detection on Small Satellites is certainly vast and would require a complete PhD thesis on its own. This chapter deals with the problem of detecting failures on components of the AOCS by using a domain of AI called Expert Systems (ES). In particular, the specific category of ES here presented is that of the Fuzzy Logics, and the actuator to which the algorithm is applied are the Magnetic Torquers (MT). The presented case study can be considered a feasibility study, but already demonstrates two results:

- The Fuzzy Logics are powerful and can be configured to perform failure detection
- The expert knowledge is effectively represented by the FL and the functioning of the algorithm represents the reasoning that the expert would perform

7.2 Reference Mission

The reference mission for the presented Case Study is a nanosatellite mission developed by the CubeSat Team at Politecnico di Torino, called 3-STAR. The main objective of this program is to provide educational and hands-on experience to the

numerous students participating in the team. The technological objective of the mission will be performing stereoscopy experiments from space, possibly testing and validating inspection algorithms to be later reused in other nanosatellites missions. In addition to the main mission objectives, 3-STAR will be used as a validation platform for different technologies currently being developed in the team's facilities. Among these, Artificial Intelligence (AI) based Autonomous Command and Data Handling System (A-C&DH) and Attitude Determination and Control System (A-ADCS) will be included [96]. The satellite is envisioned to be a 3U CubeSat, featuring a commercial bus platform developed and sold by one of the major companies (Tyvak Int., Clyde Space, GomSpace, and so on) and will likely feature as payload one or more cameras, and an in-house developed COMSYS board, either as main telecommunication unit or as a redundant one with respect to the platform one. As of June 2017, the mission and preliminary system design has just begun, thanks to the new students of the CubeSat Team. No additional information is available at the moment. Given the direction of FSW development taken in the past years, the FSW will be developed in Python, with additional libraries developed in C/C++ for performance reasons.

7.3 Fuzzy Logics Application

The FL application developed for this case study aims at detecting the failures of a specific set of actuators (the Magnetic Torquers) of a 3U CubeSat spacecraft. The purpose of the application is to demonstrate the feasibility, and similar detections can be performed on other sensors or actuators of a spacecraft, provided the failures to be detected are modelled and their behaviour is known. Generally speaking, it is possible to develop additional rules for unknown behaviours.

7.3.1 Magnetic Torquer Modelling

Magnetic Torquers are a very common and reliable actuator used to control attitude for LEO CubeSats as they are cheap, they consume a low amount of power and are typically low weight. They are typically employed in two configurations: coil (Figure 51) and rod (Figure 52).

Figure 51 Magnetic torquer example: coil configuration

Figure 52 Magnetic torquer example: rod configuration

They exploit the interaction between the Earth Magnetic Field (EMF) and the magnetic field generated by the MT.

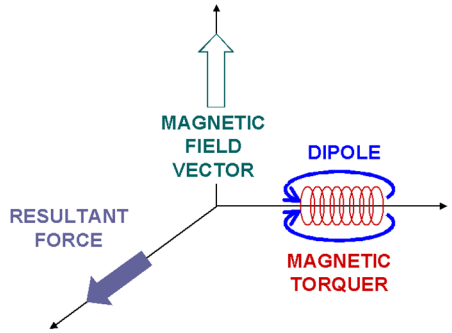


Figure 53 Representation of the resultant force due to magnetic field interaction

The interaction between the EMF and the magnetic dipole moment generates the control torque (Figure 53), and can be modelled as follows:

$$T_{control} = m^b \times B^b$$

where $T_{control}$ is the 3x1 control torque vector, m^b is the 3x1 magnetic torque dipole moment and B^b is the 3x1 EMF vector expressed in body axis.

It is possible to evaluate the dipole moment m as

$$m = NiAn_a$$

where N is the number of coils, I the current flowing in them, A the area inscribed by the coils and n_a the unit vector perpendicular to the plane of the coils. The main specification for a MT is usually the maximum dipole moment, which is a function of the number of coils, the amount and direction of the current that flows into the coils, and the area of the MT.

7.4 Failure Modelling

In order to design a Failure Detection algorithm for a certain application, the dynamics and behaviour need to be available during the design. Available is an intended vague term, because of the different approaches that can be taken, depending on whether the application involves NN or FL or other AI algorithms. In particular:

- The dynamics of the failures need to be known and modelled in order to define the rules for a FL application

- The dynamics of the failures need to be reproducible in order to correctly train a NN application

The two approaches can overlap, and in general it is considerable as a requisite to have the data concerning some example of the failures to deal with detection applications.

Despite the MT being a reliable hardware, they can be subject of failures and these events have very peculiar and recognizable characteristics. MT can fail in four different ways (Figure 54):

- *Float*: the output of the failure is zero

$$m_{float} = 0$$

- *Hard-Over (HO)*: the output assumes a ramp characteristic, until saturation value is reached

$$m_{HO} = \begin{cases} ramp & 0 < t < t_{saturation} \\ saturation & t \geq t_{saturation} \end{cases}$$

- *Lock-in-Place (LIP)*: the output is stuck to a value different than zero

$$m_{LIP} = constat$$

- *Loss of Efficiency (LOE)*: the behaviour remains similar to unaffected MT, but a lower efficiency causes the output to be reduced

$$m_{LOE} = k \cdot m_{desired}\,, \qquad 0 < k < 1$$

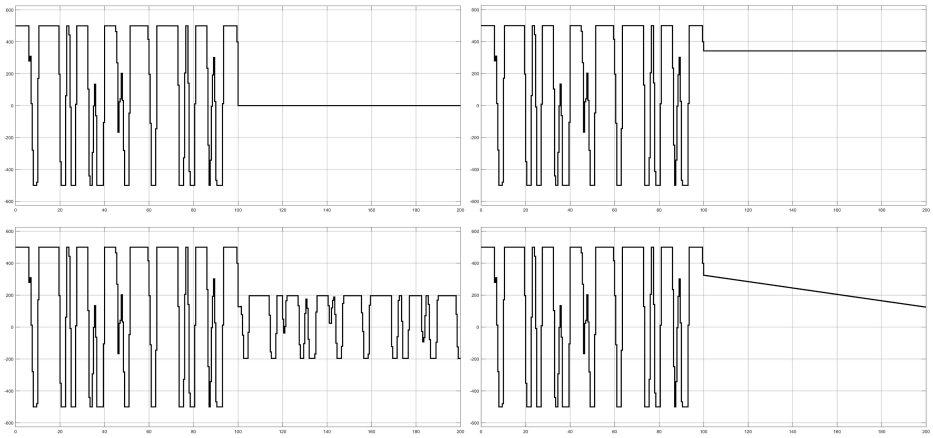


Figure 54 Failure modelling, output of the control command to the MT. Clock-wise, starting from top-left: float, lock-in-place, hard-over, loss of efficiency

The simulation of the failures was performed by setting the actuator output to match the characteristic failure. Several simulations were run, with a random initialization of the characteristic variables in order to ensure generality of the applied approaches.

7.5 Rules definition

As shown in 5.7, FL work by extending the classical logic in the continuous interval. To perform this, a set of rules and sets of input and output variables must be defined. One of the most striking characteristics of an AI application, is the fact that its behaviour can be defined, or taught, without actually coding it in the application. For ES, the knowledge of the expert involved in the design is translated into executable code.

7.5.1 Input and Output Variables and their membership functions

Five input variables were defined in the application, and are intrinsic variables that characterize the problem under analysis:

- *MT current*: the value of the current that flows into the MT. This value is straightforward to obtain, as the Analog-to-Digital Converter (ADC) current sensor is a common component

- *Derivative of MT current*: the value of the derivative of the MT current. Another straightforward value to obtain, as it can be simply obtained by sampling two consecutive times the current value
- *Double Derivative of MT current*: the double derivative of the MT current. As with the current derivative, to obtain this value two consecutive measures of the derivative of the current are needed
- *Error*: the difference between the commanded value and the measured value. Another easily obtainable value, as the commanded value is known (the controller is responsible of commanding the current value) and the measured value is known (by the ADC sensor)
- *Estimated LOE*: the value of the estimated loss of efficiency times the commanded current value minus the measured current value.

  $$estimated\ LOE = k * I_{commanded} - I_{measured}$$

  This is the less simple variable to obtain: *k* can be iteratively estimated by comparing the commanded value to the measured one. If, for example the ratio, is constant, the Estimated LOE can be obtained.

These five variables are able, along with the output variables and the corresponding rules, to define an Expert System able to correctly identify which type of failure is present on the torquers.

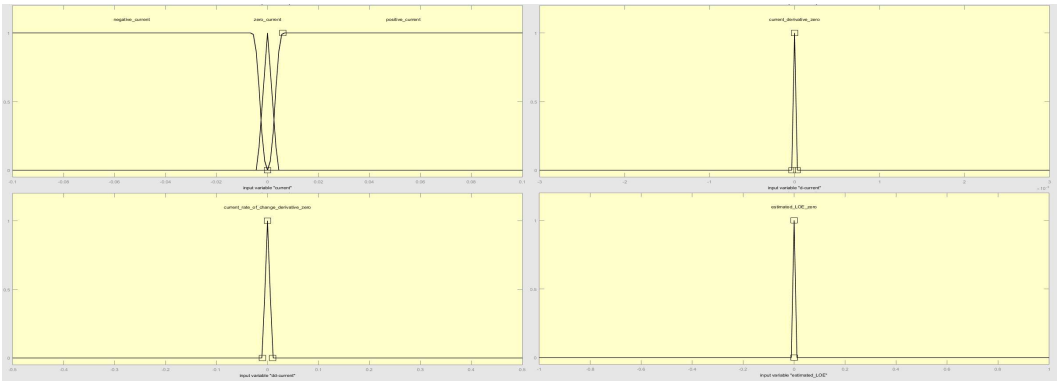


Figure 55 Input variables and their membership functions

For the presented five variables, appropriate membership functions (Figure 55) must be defined for the evaluation of the input variables. In general, for this application, some soft constraints can be guessed by the domain expert, by iterative reasoning about the dynamics of the problem. Output variables, for this particular application, do not need to be de-fuzzified (Figure 56).

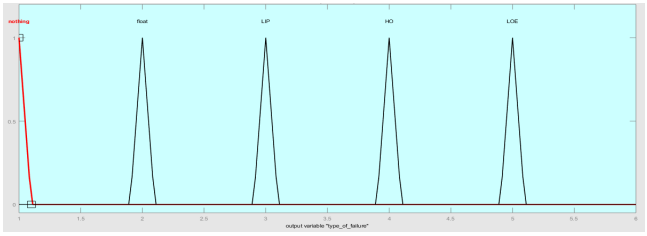


Figure 56 Output variables: de-fuzzification is not needed, as the failure identifier is an integer number

The following membership rules have been defined:

- *current*: negative (less than -0.01), zero (between -0.01 and 0.01), positive (greater than 0.01) (Figure 57)
- *current derivative*: monitored only when zero (between -0.00003 and 0.00003)
- *error*: monitored only when zero (between -0.2 and 0.2)
- *current second derivative*: monitored only when zero (between -0.01 and 0.01)
- *estimated LOE*: monitored only when zero (between -0.00002 and 0.00002)

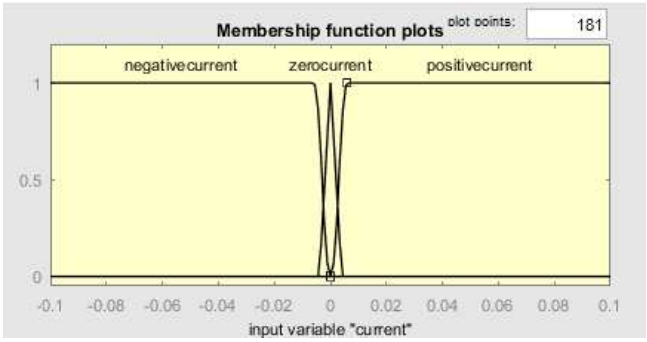


Figure 57 Membership function for the current input variable

7.5.2 Rules

The following paragraphs emulate a possible way to define the membership functions and the subsequent rules.

For a Hard-Over failure, that is constituted by a linear trend of the current value, the derivative of the current is constant. Since each HO failure can be characterized by a different constant value of the derivative, this particular variable is not

meaningful. Continuing, since the derivative is constant, the second derivative must be zero. This reasoning is meaningful: it means that the fuzzy set will have to monitor the second derivative and to be able to distinguish between a zero and a non-zero value. A possible rule can also be defined: if the second derivative is not zero, then the failure can probably be a LOE (where the current value and their derivative is changing over time).

After this reasoning, the Hard-Over behaviour is still undefined: the second derivative must be zero, but this is not sufficient to correctly identify the HO. Another rule defined for the HO is obtained by checking that the value of the current derivative is *not* zero. If it is zero, then we would be in presence of a Lock in Place failure (derivative being zero means the output current is constant).

Continuing with these types of reasoning lead to a set of rules and a set of membership functions that completely represent the expert knowledge on the problem in a computational form.

With just a set of five rules, the complete set of failures of the MT can be detected. The rules are:

- if the *current* is zero AND the *current derivative* is zero AND the *error* is NOT zero AND the *estimated LOE* is not zero then failure is *float*
- if *current* is NOT zero AND the *current derivative* is zero AND the *error* is NOT zero AND the *estimated LOE* is NOT zero then the failure is *lock in place*
- if *current derivative* is NOT zero AND the *error* is NOT zero AND the *current second derivative* is zero AND the *estimated LOE* is NOT zero then failure is *hard-over*
- if the *error* is NOT zero AND *estimated LOE* is zero then failure is *loss of efficiency*
- if the *error* is zero then NO failure is present

It has to be noted that, for this specific case study, the number and the complexity of the rules is low: for different applications, more complex and more numerous rules can be expected.

7.6 Results

At each sampling step of the on-board software, it is possible to obtain all the five input variables (except for the starting steps where no derivatives exist), and at each step it is possible to evaluate all the defined rules in the system (Figure 58).

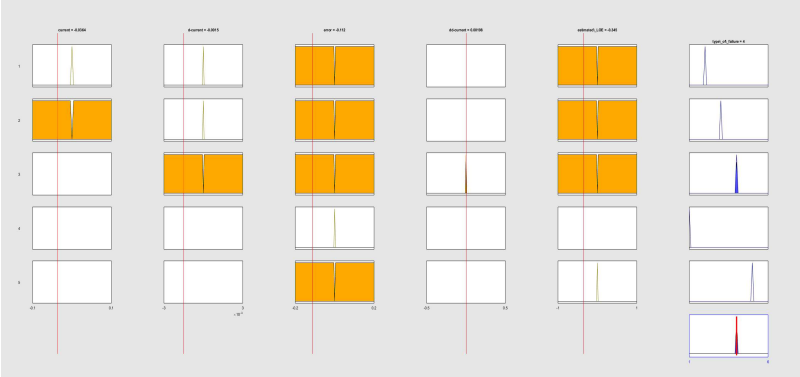


Figure 58 Rule evaluation and failure detection: hard-over detected

Simulations that iterate the appearance of failures in the system (injecting the failure by overring the output of the current sensor), and the behaviour of the detection with the FL has been evaluated. The system is able to correctly identify all failures. Instabilities are present in the final output and they are due to the fact that sometimes, for certain values and certain types of failures, the commanded value is coincident with the measured (faulty) one. This causes instantaneous shifts to the status of *no failure*, and therefore instabilities in the detection. However, this is not an issue, as the detection of a failure can take place in several steps, and therefore basic filtering can be applied. A typical FSW runs at a speed of 2Hz or more, therefore commanding a different value of the current at each step: filtering over a period of a couple of seconds does not alter the quality of the detection and allows to remove the instabilities due to the phenomenon described earlier (Figure 59).

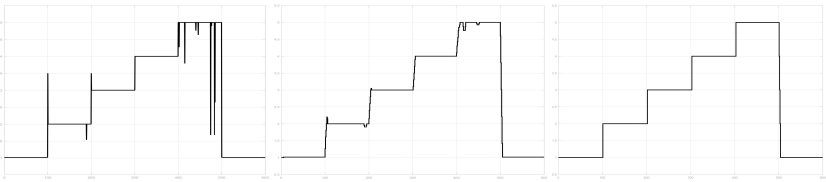


Figure 59 Output of the Expert System: from the left, unfiltered, basic and medium filters applied. Each step represents a different value of the output variables, therefore represents a different failure detected

7.6.1 Review

The applications presented in this chapter deal with a well-known domain of spacecraft engineering: failure detection. Several algorithms categories can be used to solve the problem of detecting failures in actuators and sensors. Neural Networks are an example of algorithm category that can be used. A first conclusion that can be drawn when comparing NN with ES, is the increased computational cost of NN.

The following table summarizes the capabilities of Expert Systems to perform Failure Detection.

Table 15 Summary of ES algorithms characteristics when applied to FDIR

Review Parameter	Comments
Benefits	**Knowledge Implementation** – The knowledge transfer from an operator to the program can be performed in a structured way, without hard-coded programming of the behaviour of the system. **Performances** – Simple ES obtain high detection rates even for complex problems such as failure detection. **Computational Costs** – With respect to other algorithm domains (such as ANN), ES are able to reach high detection rates by requiring considerably smaller computational costs.
Limitations	**Scalability** – ES implemented via FL are ideal for small problems, such as actuator monitoring. Increasing the architecture of the detection problem, the number of rules can considerably increase. Other types of ES need to be considered in this case.
Applicability	**Scope** - Applicability of ES is vast, and applications are appearing in many engineering domains.

Chapter 8

Case Study: Tradespace Exploration with Genetic Algorithms

8.1 Background

The purpose of multi-attribute tradespace exploration is to capture decision-makers preferences and use them to generate and evaluate a multitude of system designs, while providing a common metric described in a stakeholder friendly language. To achieve this, the Multi Attribute Utility Theory (MAUT) is employed for the aggregation of the preferences from all the stakeholders. MAUT is widely used in the fields of economics, decision analysis, and operational research. It postulates that people make decisions based on value estimates of personally-chosen reference outcomes. Decision-makers interpret each outcome in terms of some internal reflected value, or utility, and they act in order to maximize it. In the case of multiple attributes, an elegant and simple extension of the single attribute utility process can be used to calculate the overall utility of multiple attributes and their utility functions [97], [98]. There are two key assumptions for using this approach:

- *Preferential independence*, that means the ranking preference of a pair of attributes is independent with respect to the other ones

- *Utility independence*, or the independence of preference intensity, that means that the "shape" (shown in Figure 60) of the utility function of a single attribute is independent of the value of the other attributes.

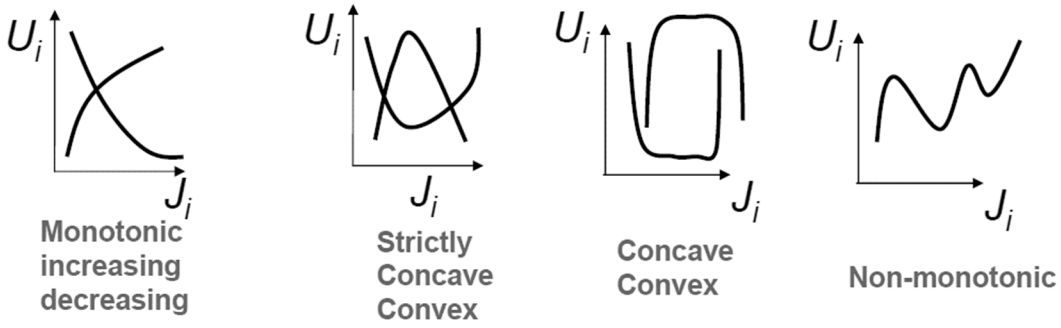


Figure 60 A few examples of utility function. Credits MIT

The non-linear behaviour of the utility functions is strongly related to the uncertainties of the outcomes of the decision process. This is caused by the non-linear evaluation of benefits and by the experts' attitude with respect to risky scenarios. If the above assumptions are satisfied, then the multiplicative utility function can be used to aggregate the single attribute utility functions into a combined function according to

$$\boldsymbol{KU(X) = \prod_{i=1}^{N} Kk_i U_i(X_i) + 1} \quad (1)$$

- K is the solution to $K + 1 = \prod_{i=1}^{N} Kk_i + 1$ and $-1 < K < 1\ K \neq 0$
- $U(X), U_i(X_I)$ are the multi-attribute and single attribute utility functions, respectively.
- N is the number of attributes.
- k_i is the multi-attribute scaling factor from the utility interview

The values of each k_i give a good indication of the importance of each attribute (i.e. a kind of weighted ranking) and are bounded between 0 and 1. The scalar K is a normalization constant that ensures the multi-attribute utility function has a zero to one scale [99]. Despite the attractiveness of an axiomatically-based decision model, empirical evidence shows that people do not obey expected utility theory in daily decision-making due to systematic biases in their thinking. For this reason, the logic flow of the method involves the definition of stakeholder attributes,

context variables and design variables [100]. Once those elements are defined is possible to develop system performance and value models, aiming to evaluate the multi attribute utility and the costs involved in the project life cycle.

When applying the MAUT to a particular problem, the effects on the utility given by the different attributes are highlighted. In this case, a Multi Attribute Tradespace Exploration (MATE) analysis is obtained. Given the complexity and the variety of different possible choices during the conceptual phase of a space mission, this technique is particularly suitable for assuring that all the various options have been considered, including programmatic and technical aspects, such as manufacturability, assembly, operations, and physical architecture choices.

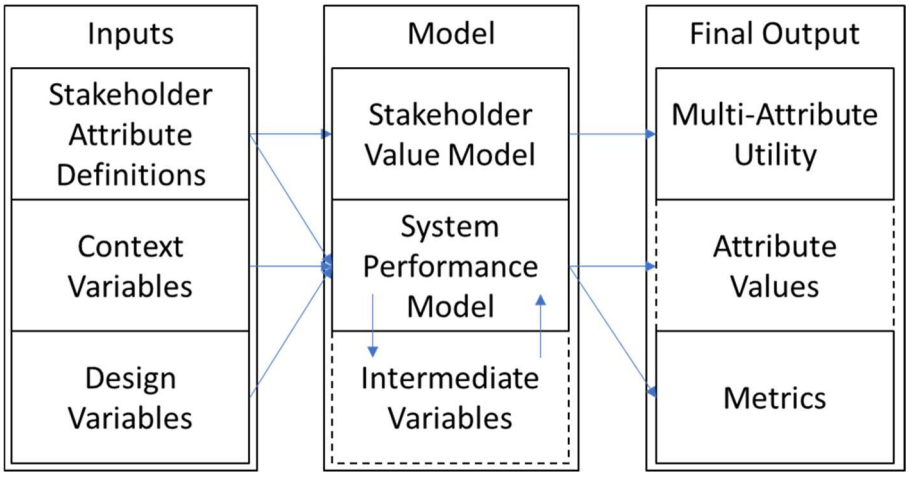


Figure 61 MATE logic flow

Once all the aspects involved in the MATE are defined, it is possible to develop a code which automatically explores the tradespace and gives as final output the best choices with respect to all the involved stakeholders needs (Figure 61). Several options exist to explore the tradespace: depending on the analysis models and design variables, a specific exploration methodology may be required.

In literature, several applicable exploration methodologies have been studied: complete exploration of all the problem solutions, optimization with Simulated Annealing techniques [101], Normal-Boundary Intersection [102], Nelder-Mead Simplex [103], Artificial Intelligence with Particle Swarm Optimization [104], and Genetic Algorithms [105]. It is evident that a bigger size of the tradespace requires a guided exploration to avoid excessive computational effort and avoid any loss of solution candidates.

8.2 Reference Mission

The trade-off capabilities of the MATE methodology and the exploration powers of GA show very promising results when applied to the design of CubeSat missions, especially thanks to the peculiarities of the CubeSat standardized design. The reference mission considered in the research presented consists of one or more CubeSats employed as secondary payloads of a flagship mission. They will be deployed during mission operations in situ. The objectives of the CubeSat mission are to provide scientific support to the mothership, either by repeating one or more of the main spacecraft's measurements, or by supporting the science goals by performing additional measurements. In addition, the CubeSats can also perform technological demonstration. Since the CubeSats are secondary payloads, constraints imposed by the flagship mission have been considered: maximum occupied volume, maximum single satellite size and weight, specified interfaces and operational requirements. A space mission concept that can be cited as a reference is the CubeSat Opportunity Payload Intersatellite Network Sensor (COPINS) mission [106]. The mission is the same as that considered in Chapter 6.2.

When considering the conceptual design of similar missions, it is evident that several different architectures and systems designs are possible, and they can all potentially satisfy the stakeholders of the mission. For example, a CubeSat mission composed of 6 single unit CubeSats could provide similar results to a mission composed of two 3-unit CubeSats, depending on the design. This is because the only volume requirement considered in the case study is that the CubeSats shall occupy a total of 6 units, with dimensions of a single satellite up to 3 units. The same concept applies to other characteristics of the mission and the system, such as the mission timeline, the scenario, the mission phases, operation strategies and more. It is evident how a methodological approach should be used in exploring all the different design choices, in order to come up with a mission baseline that provides the best utility and biggest contribution to the results of the main mission. For this reason, a Matlab(®)/Simulink(®) algorithm has been designed to explore all the different mission architectures and concepts of operations that can be generated. In particular, the solutions generated by the algorithm should represent as closely as possible a complete mission concept. With this objective in mind, the computational problem becomes complex, due to the presence of a high number of design variables and the selection of components available. The design

vector dimension can reach sizes of more than 30 design variables, adding up to a solution space in the order of billions of different architectures. It is therefore unfeasible to evaluate all the possible solutions [107].

8.3 Genetic Algorithms for Tradespace Exploration

In real world applications, most of the optimization problems involve more than one objective to be optimized. These objectives are often conflicting, i.e., maximize performance, minimize cost and maximize reliability. When this happens, a single extreme solution would not necessarily satisfy all the objective functions and the optimal solution for one objective may not be the best solution for other objectives. Therefore, different solutions will produce trade-offs between objectives and a set of solutions is required to represent the optimal solutions group. The trade-off curve reveals that considering the extreme optimal of one objective (for example, costs) might require a compromise in other objectives (for example, spacecraft reliability). The solution to this problem can be found among the pareto-optimals. A pareto-optimal is an optimal solution with respect to all objectives that cannot be improved in any objective without worsening another one. The set of all feasible solutions that are non-dominated by any other solution is called the pareto-optimal or non-dominated set; the values of objective functions related to each solution of a pareto-optimal set, evaluated in the objective space, is called pareto-front.

8.3.1 Intelligent exploration

In complex problems, such as the conceptual design of a space mission can be, the number of solutions that form the design space can reach. It is therefore mandatory to exploit structured and efficient ways to explore the design space and evaluate the solutions, in order to keep the computational cost and the exploration duration acceptable. Depending on how the problem is constructed in the first place, several different exploration methods exist, that can move through the space both in case of a continuous space and in the case of a discrete one. Examples of these methods can be genetic algorithms for discrete problems, or simulated annealing for continuous ones [73], [108], [109]. The present work explores the use of genetic algorithms (GA), performing an exploration type called guided random search [110]. These types of algorithms are inspired by the selection process of nature, which causes the stronger individuals to survive in a competitive environment. In nature, each member of a population competes for food, water and territory, and also strives for attracting a mate. It is obvious that the stronger individuals have a better chance for reproduction and creating offspring, while the weaker performers

have lesser chances of producing offspring. Consequently, the ratio of the strong or fit individuals will increase in the population, and overall, the average fitness of the population will evolve over time. Offspring created by two fit individuals (parents) has a potential to have a better fitness compared to both parents: the resulting individual is called super-fit offspring. By this principle, the initial population evolves to a better suited population to their environment in each generation [111].

8.3.2 Population dynamics

In genetic algorithms, each solution of the problem is represented by a set of parameters known as genes, and these are joined together in a genome. A genome, which describes an individual, evolves through iterations called generations. The dynamics of each individual inside the population are ruled by a function that evaluates how well the considered individual performs in the environment it is in. The mentioned function is called fitness or objective function. Finally, during the various iterations, a selection of the parents for reproduction and recombination is applied [112]. The main objective of selection operator is to pick the fit solutions and eliminate the weak individuals. In the reproduction phase, the two parents identified by the selection operator recombine to create one or more offspring with the crossover operator. There are several different crossover operators in the literature, although the underlying mechanics is similar: selecting two strings chromosomes from the mating pool and exchanging some portion of these two strings in order to create new individuals. The purpose of this operator is to perform a rapid exploration of the search space. Another operator that can be applied is the mutation operator. It is applied to individual solutions after reproduction: one or more genes are randomly changed in an individual, after a selection has been applied. The mutation operator usually affects small portions of the population. The aim of this operator is to maintain the diversity of the population and to increase the possibility of finding the global optimum. To sum up, the selection operator selects and maintains the good solutions; the crossover recombines the fit solutions to create fitter offspring and the mutation operator randomly alters one or more genes in the offspring with the intent of maintain the evolution dynamic. The next section will cover in details the problem setup: in particular, the characteristics of the individuals will be described, highlighting how these form the Design Vector (DV), and how the genetics algorithms are employed to explore the tradespace.

8.4 Algorithm Design

8.4.1 Architecture

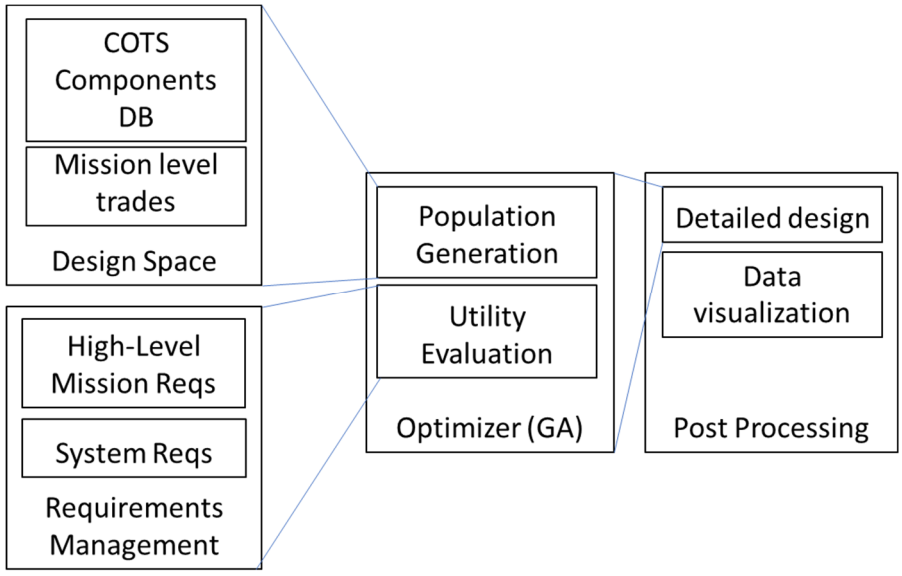


Figure 62 The implemented algorithm consists in combining Genetic Algorithms with Multi-Attribute Tradespace Exploration. Solution generation, requirements management and post-processing design and visualization are also performed.

Figure *62* shows the architecture of the implemented solution. At each algorithm iteration, a population of individuals is selected and evaluated. As specified earlier, each individual carries information concerning mission architecture aspects, system design and components. Once the current population is generated, the fitness of each individual is evaluated: this can be thought as evaluating the utility of the corresponding mission concept. During the utility evaluation, high-level requirements are also verified, and the individuals that violate any requirements are penalized, receiving an utility score of zero. By design, GA select the most fit individuals by using a tournament selection: this process guarantees the correct elimination of the individual that violates the requirements and of the unfit individuals. Finally, once the optimization has selected the most fit individuals, additional post-processing algorithms are executed, to finalize the design and to generate data products comparable with those generated during a CD session by the domain experts.

8.4.2 The Design Vector

The DV is the vector that describes a specific solution and that contains all the information needed to define a particular mission concept. It is composed of 36 variables that store information about several aspects of mission architecture and system design.

The DV structure was defined by analysing the mission goals and by selecting both mission and system technical domains that are critical during the preliminary design of a space mission.

Table 16 Design Vector attributes categories

Design Vector Categories	Parameters	Equipment	Number of Parameters
Autonomy	Goal definition, event reaction, data selection, knowledge from measurements, failure detection, isolation and recovery	-	6
Communication Architecture	Percentage of data rate used, number of antennae, Earth communication	Radio, Antenna	5
On Board Processing	Command and Data Handling architecture, radiation tolerance	Processor Family	3
Primary Payload	Camera technology, spatial resolution, optics volume, maximum frames per second, number of sensors	Camera	6
Secondary Payload	-	Any in the databases	1

Guidance, Navigation, Control	Trajectory planning, attitude determination performance, attitude control performance, position determination performance, position control performance	Sensors and actuators	7
Data Acquisition	Data acquisition strategy	-	1
Operations	Lifetime, mothership/daughtership interactions	-	2
Orbit Architecture	Altitude, inclination, formation flying, constellation	-	4
CubeSat Number	Number of CubeSats considered	-	1
Objectives Accomplished	Each scientific / technological objectives	-	n

Table 16 shows a summary of all the categories that were included in the DV. The first column shows the category, while the second and third one list all the different parameters that were included in each category. Finally, the last column condenses the information in a number, which represents the total number of parameters for each category.

8.4.3 The Algorithm

The key part of the research relies on the algorithm that, from the definition of the DV, creates each solution during the exploration.

The approach used involves GA to solve an integer problem: each parameter in the DV is associated to an integer that represents the number of alternatives for the specified parameter. The number of possible alternatives is defined by each domain expert. For example, the *event reaction* parameter in the *Autonomy* category has a value of 3 associated with it: this means it can assume three different configurations,

as specified by the Flight Software Engineer: no event reaction is planned; meaningful events are detected and then mission control is informed; meaningful events are detected and mission re-planning is executed.

For the parameters in the third column, the approach is similar but each parameter corresponds directly to an equipment category. For this, a CubeSat component database was implemented. Four mandatory parameters were included for each component in the database: mass, power, cost and size. Other parameters were added, and are especially useful since they can be later used to verify the compliance to the requirements, or to compute the fitness value. For example, the *Camera* parameter can assume a value from 1 to 4 that corresponds to a specific COTS equipment found in the database: a CMOS camera; a basic spectrometer; a high-performance spectrometer; a CCD camera.

Custom population creation, cross-over and mutation functions were designed to support the presented setup. Creation function initializes every individual of the population, picking a random integer value constrained from 1 to the maximum value for each DV variable. Single point crossover has been chosen as crossover function. Mutation function affects only a small number of individuals in the generation, and for these, only one gene is re-initialized to a random value, as constrained by the DV.

Lastly, the fitness of each individual of the population is computed using (1).

8.4.4 The optimizer

As introduced earlier, genetic algorithms are the key technology used to explore the tradespace. The configuration of the algorithm was as follows: initial population was set at 540 individuals, crossover fraction was set at 0.95 (meaning that to the remaining 0.05 the mutation operator was applied) and the elite population fraction was set at 0.35 (Figure 63). The selection was made with tournaments. Infeasible solutions, for example those that violated the requirements, were discarded and the population was re-initialized randomly for the removed individuals.

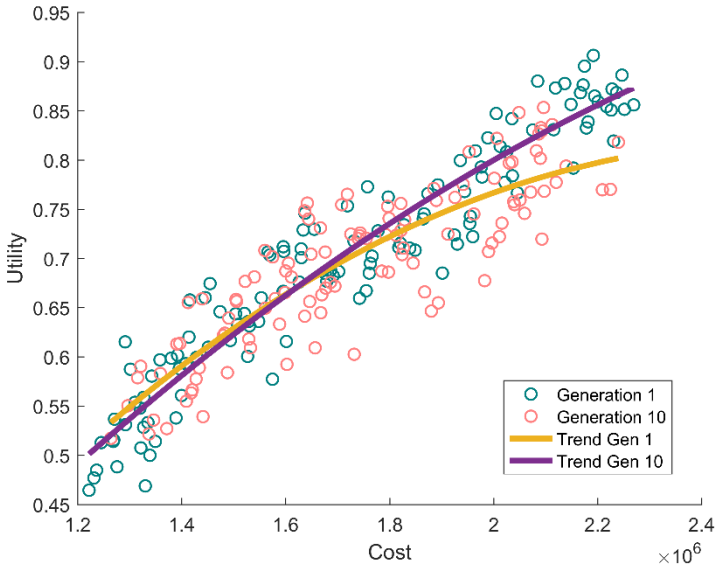


Figure 63 Optimization process: evolution in time of the population. The improvement of the utility with the increase of the generation number is shown.

Several configurations have been tried, since optimal initial configurations for GA are highly dependent on the problem analysed. The details are summarized in Table 17: the table shows both the final values selected for the simulations, and the ranges that were used when defining the optimal values.

Table 17 - Genetic Algorithms configuration parameters

Configuration parameters	Value	Explored values
Population size	540 individuals	180, 360, 540, 720, 1000
Crossover fraction	0.95	0.75, 0.9, 0.95, 0.99, 1
Mutation fraction	1 – *crossover fr.*	0, 0.01, 0.05, 0.1, 0.25

Elite population fraction (paretofraction)	0.35	0.1, 0.35, 0.5
Selection	Tournament	-
Requirement violation approach	Individual removal	-

Population size has been chosen to be 15 times the number of variables in the DV, as a balance between smaller populations (increase in the convergence speed) and bigger ones (higher chances of having more optimal solutions in the initial population) [113]. *Crossover fraction* was chosen at 0.95: this choice resulted in a greater effect of the reproduction dynamics with respect to the mutation ones. *Mutation fraction* was chosen to be 0.05, thus applying the mutation function only to the population that did not reproduce. *Elite population* was set at 0.35, meaning that the 35% of the new generation is formed by individuals picked from the old generation. The selected value ensures a balance between effectiveness of the search (lower elite population fractions) and survival of fit individuals (higher elite population fractions).

8.5 Results

The investigation on methodologies to improve and automate the space mission and spacecraft design is a vast effort, branching out into many fields of science and engineering. The proposed research obtains several important results towards the design of space missions that provide higher utility to the stakeholders, by being more optimized and not bound to the stagnancy of conservative mission design approaches. These improvements are obtained through innovations in three aspects of the mission design:

- exploring the alternative concepts thoroughly and more efficiently thanks to the MATE and GA approach
- considering the availability of certain highly standardized components thanks to the component database included in the algorithm architecture
- ensuring effective final solutions that comply with high-level requirements

Furthermore, domain experts and mission designers obtain significant improvement to the mission design process, thanks to the decision-making support and the post-processing algorithms that emulate CD sessions.

8.5.1 Efficient tradespace exploration

Depending on the dimension of the design vector and the ranges of the considered variables, the number of solutions forming the tradespace can well surpass the order of billions. In the presented case, 36 variables add up to more than 10^{17} different solutions. When, for each solution, a utility function must be evaluated, it is evident that the problem becomes computationally expensive.

The use of guided random search strategies, implemented with GA, allows the exploration and the discovery of the optimal solutions without evaluating the fitness function for all the individuals, but only for a restricted set. Figure 64 shows several plots of a limited set of the solution space for this problem, that give a glimpse of the shape of the whole tradespace. As shown in the figure, the MATE and GA implementation optimizes the search to define the pareto front for the analysed problem.

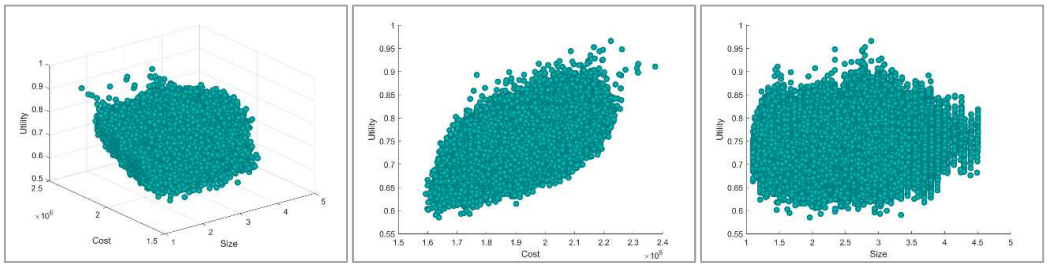


Figure 64 Solution spaces (100k points): from the left, cost-size-utility, size-utility and cost-utility plots

8.5.2 Impact of the CubeSat database integration

The integration of a component database in the architecture infuses the obtained solutions with information regarding parameters such as power consumption, sizes, performances and so on. The knowledge on these parameters would traditionally be acquired later during the design process.

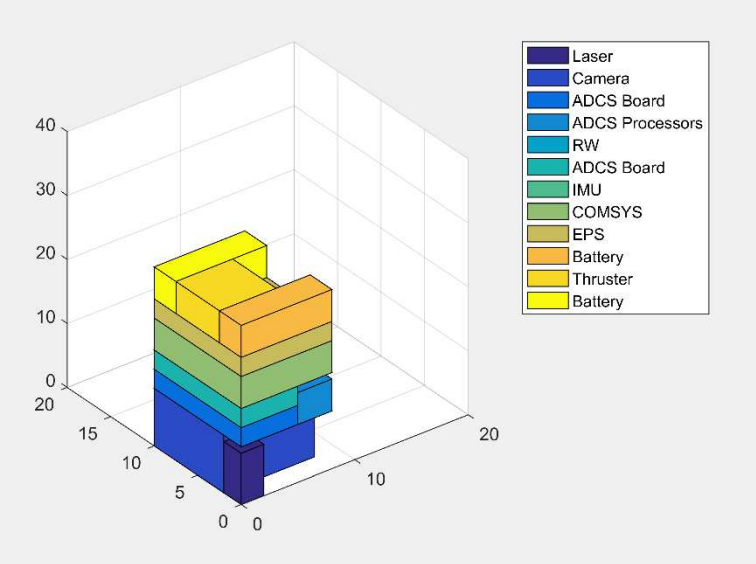


Figure 65 3U internal configuration

With this approach, instead, as the GA creates new individuals, it defines specific hardware configurations. This enables interesting analysis once the solution is selected in the final iteration. The possibilities opened by this implementation are numerous, and here the most promising ones are reported:

- mission and system budget definition (mass, link, power, delta-V) thanks to the definition of the component list and mission architecture
- optimization of the internal configuration: the component list includes information on volumes and specific component requirements, such as positioning inside the spacecraft (Figure 65 shows a 3U CubeSat configuration obtained by the algorithm)
- detailed design: by defining a power budget and a list of operative modes, the solar panel and battery sizing can be automatically computed (Figure 66 shows examples of solar panel design)

8.5.3 Requirements compliance

Thanks to the capability of the GA optimization to handle DV composed by a high number of variables, it is possible to increase the number of variables representing mission and system level aspects, directly matching them with high-level mission and system requirements. This approach ensures that the corresponding design produced by the algorithm is compliant with the requirements specified. This is done by setting the solution individual fitness value to zero if the one or more

requirement is not met. In this way, selection dynamics will remove the unwanted solution.

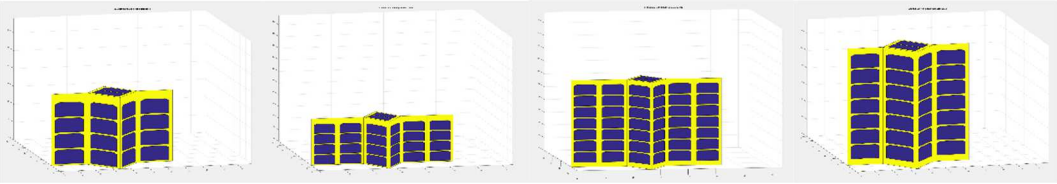


Figure 66 Solar panels configuration: example outputs

Another important but less evident result of enforced requirement compliancy is related to the biased attitude of human experts towards computer generated solutions, especially when artificial intelligence is involved. In this way, the obtained solutions are more likely to be accepted by the engineers involved in the early design phase.

8.5.4 Algorithm performance comparisons

Table 18 - Algorithm performance comparison

Algorithm	Problem Considered	Time Complexity	Average Execution Time	Pareto Front Found
Monte Carlo	$8*10^{16}$	O(N*f)	Undefined	No
GA	$8*10^{16}$	O(n*G*f)	1 hour	Yes
Non-Guided Exploration	$8*10^{16}$	O(N*f)	$1.8*10^{10}$ hours	Yes
CD	Inherently smaller	-	1-2 weeks	No

The application of MATE analysis to engineering problems requires the implementation of an explorer that navigates the solution space, and depending on the dimension of the problem and the explorer design, obtaining the pareto front can be expensive, both from a computational cost and time perspective. Table 18 presents a comparison with other methods used to explore a tradespace. The algorithms explored are: Monte Carlo method, GA, Non-Guided exploration (where every single solution of the tradespace is evaluated) and CD. The CD approach is reported for additional comparison with traditional methodologies for space mission preliminary design. The tradespace size is also presented to offer a comprehensive view of the comparison. In particular, for CD sessions, the solution space defined by experts is smaller than the one implemented on a computer simulation: considered solutions are biased towards previous experience, preferences of the experts, adversity towards innovation and bias towards safer solutions. Moreover, not only the generation of a proper tradespace is challenging for a human expert, but this space will be biased towards the preferences of the expert himself, instead of reflecting the stakeholders' goals. Time Complexity column describes the complexity of the algorithm from an execution time perspective, using the common Big O representation. N represents the solution space size, f the complexity of the fitness function, n the max number of generations for the GA, and G the GA population size.

8.5.5 Review

The application presented in this chapter explores the concept of autonomy in a different way: the use of AI to solve the problem of preliminary mission design, and in particular to quickly and efficiently explore the set of possible alternatives to mission design that can be generated from the stakeholder analysis. In addition, a clear benefit of implementing AI in this type of problems, is that the solutions are generated without traditional biases that would affect human designers. The evaluation of solutions is also performed considerably quicker with respect to traditional preliminary design generation. The following table summarizes the capabilities of GA to perform Multi-Attribute Tradespace Exploration.

Table 19 Summary of GA algorithms characteristics when applied to MATE

Review Parameter	Comments
Benefits	**Unbiased exploration** - solutions are discovered and analysed without interference with previous knowledge or methodologies, even those that would be hardly detectable by human operators. **Analysis speed** – solutions are processed and analysed much faster with respect to a human operator. **Traceability** – solutions are directly originated and evaluated from the stakeholders' needs
Limitations	**Discreet optimization** – solutions are defined as vectors of integers: in this perspective, dealing with continuous problems requires a modified approach
Applicability	**Scope** – GA as engine for performing MATE can be employed not just in the space mission analysis domain, but in other fields of engineering.

Chapter 9

Conclusions

The thesis presents the results of three years of PhD research on Mission Autonomy for Small Satellite missions. In particular, the key focus of the research was exploring the capabilities and potentialities of Artificial Intelligence to innovate and improve the autonomy level of the future missions, both interplanetary and Earth orbiting. Several reasons motivate the selection of the domain, the methods, and the case studies, and they can be understood considering the background of the research group this research was carried out in.

The domain: Small Satellites

CubeSats were born in 1999 as an educational tool, to ease the process with which students could acquire spacecraft engineering experience and perform space-related practical research. Now, in 2017, after almost 20 years, CubeSats have definitely evolved towards becoming a fully capable space systems category: scientific, technological and innovative missions are now designed with CubeSats playing the main role. Small Satellites, the bigger counterpart of CubeSats, have somehow lead the way, thanks to an easier transition from the world of flagship, expensive and performant spacecraft, to the world of miniaturized, multiple and flexible ones. After 20 years, the overall picture of the health and status of the technology is clear: spectacular adoption rates, world-wide participation with spacecraft developed and launched by many countries (of some of them, CubeSats represented the first and only affordable and feasible way to start a national space program). Small Satellites have, from the beginning, always been characterized by

solid organizations of the industries involved in their design and development, probably thanks to an easier adaptation of standard procedures and methods to the smaller class of spacecraft. CubeSats, on the other hand, have experienced tough problems due to inherently less experienced players involved: reduced reliability, lower-quality components used, more agile and less controlled development processes, are all causes of the sustained failure rates for this type of technology. Moreover, several problematic points have, since the beginning, affected the spacecraft category and impeded a complete adoption: the slow evolution and improvement of telecommunication systems, propulsion systems and overall materials and components have played a big role in stopping some interesting concepts from becoming a reality in the early years.

In the last decade, the panorama has changed: technology has evolved, and more daring missions have been proposed and are now under development, with improved payloads, communication technologies and propulsion systems. For these missions, the CubeSat standard and, in general, the modified approach to small spacecraft and mission design, have a noticeable effect most of the domains involved. One key area is left behind: operations do not seem to scale by scaling the technology, and little effort has been spent into disrupting and innovating how operations are designed and managed for small satellite missions. Nevertheless, Small Satellites platforms are the best candidate to demonstrate new concepts for mission operations, as they possess the required flexibility and they welcome innovative technologies (even if with a suboptimal TRL). Moreover, the category of small satellites was selected thanks to a higher average computational capability and to development approaches more comparable with traditional embedded approaches.

The focus: Mission Autonomy

The presented work focused on improving the operation architecture and management of Small Satellite missions, both Earth based and interplanetary. The main reason for this choice is that operations have not been object of extensive research such as other areas in a small satellite mission, and there have been many possibilities of improvement. Among the operations, focusing on Mission Autonomy was a straightforward choice, as the state of the art is currently aiming at streamlining operations design around the highest possible level of autonomy, as specified by ECSS. To date, very few examples of autonomous spacecraft have

flown. The present work, in addition, aims at raising awareness on the topic of Mission Autonomy and innovative operations design.

The proposed algorithms described in the thesis bring many advantages, impacting different segments of the mission architecture. As far as the space segment is concerned:

- Autonomous Event Detection allows for the design of complex operations during the mission. Furthermore, payload data downlink will benefit thanks to the fact that only the highest priority images are selected and sent to Earth, reducing the quantity of downlinked data and improving its quality. In the ground segment, a reduced and improved data flow allows for more agile resource allocations. These advantages are mission specific, but could be easily generalised for other applications
- Intelligent Failure Detection, Isolation and Recovery is another step in the direction of more reliable, performing and autonomous missions. As with Event Detection, increasing the performance of failure detection system could enable not only more efficient ground operations (as the operators are supported in taking decisions concerning failures) but also to enable innovative recovery actions or to exploit the system capabilities to fail operationally

The last case study presented aims at improving another area of mission design and development: the preliminary design:

- Supporting decision makers in their activities (be them mission design, or operations) is certainly welcomed. One of the key area where the presented thesis focused was on the design of small satellite missions, and in particular on automating tasks that are currently performed by domain experts, such as component database search and spacecraft configuration assessment. Given the standardization available for this category of spacecraft, the autonomation potential in the preliminary design phase is extremely high, and the result of this effort is that less errors affect the design of a mission, especially in a phase where the uncertainty about the system is high. Costs will also benefit from this automation, as correcting design errors further down the design process is costly and not efficient.

The technology: Artificial Intelligence

Artificial Intelligence is certainly a hot topic in research in these years: applications in medicine, image recognition, security, natural language processing, and more, are appearing and they are drastically changing the way we approach and solve problems. Most importantly, they are performing pretty well and the future improvements are promising. Space engineering is not immune to the diffusion of AI, and the research is embracing AI for several different applications, from failure detection and prognosis, to mission replanning, to spacecraft design, to payload data processing and big data analysis, and the list continues. The thesis, and the related research performed, wanted to serve as a first effort in exploring the capabilities of AI for several different applications. The results presented in the case study chapters are promising: applications compatible with the capabilities of Small Satellites can be developed and they greatly improve the way missions are managed, resulting in faster mission success and more reliable mission operations. Among the case studies presented, AI algorithms were developed reusing known literature, but an adaptation of the methodologies had to be envisioned to make the technology suitable for a space mission, especially from the flight software point of view. The innovative training algorithm developed under this research is an example of adaptation that was necessary, yet that produced promising results.

In conclusion, it has been proven that the proposed applications and methodologies are effective in improving the management and the design of Small Satellite mission operations, and that the presented case studies can be adapted both for Earth orbiting and for interplanetary missions. Future space missions will make extensive use of Artificial Intelligence, and the thesis aims at being one of the first step in that direction.

References

[1] R. Sandau, “Status and trends of small satellite missions for Earth observation,” *Acta Astronaut.*, vol. 66, no. 1, pp. 1–12, 2010.

[2] NASA, “Small Spacecraft Technology State of the Art,” no. February, pp. 1–197, 2014.

[3] J. R. Wertz, D. F. Everett, and J. J. Puschell, *Space mission engineering : the new SMAD*. Microcosm Press, 2011.

[4] R. Mozzillo, “Technologies and methodologies for CubeSat performances improvement,” Politecnico di Torino, 2016.

[5] A. Babuscia *et al.*, “CommCube 1 and 2: A CubeSat series of missions to enhance communication capabilities for CubeSat,” *IEEE Aerosp. Conf. Proc.*, 2013.

[6] J. Schoolcraft, A. Klesh, and T. Werne, “MarCO : Interplanetary Mission Development on a CubeSat,” in *AIAA SpaceOps Conference*, 2016, pp. 1–8.

[7] Planet Labs, “Planet Labs Specifications : Spacecraft Operations & Ground Systems,” 2015.

[8] Oerlikon Space, “The Optel 02 Model Optel 02 Terminal Specifications.”

[9] “Fly Your Satellite! CubeSats phoned home / CubeSats - Fly Your Satellite! / Education / ESA mobile.” [Online]. Available: http://m.esa.int/Education/CubeSats_-_Fly_Your_Satellite/Fly_Your_Satellite!_CubeSats_phoned_home. [Accessed: 05-Jun-2017].

[10] “CubeSats - Fly Your Satellite! / Education / ESA.” [Online]. Available: http://www.esa.int/Education/CubeSats_-_Fly_Your_Satellite. [Accessed: 05-Jun-2017].

[11] A. Heiney, “Project ELaNa: Launching Education into Space,” 2015. [Online]. Available: https://www.nasa.gov/content/about-elana. [Accessed: 05-Jun-2017].

[12] E. Mahoney, “NASA’s CubeSat Launch Initiative,” 2015. [Online].

Available:
https://www.nasa.gov/directorates/heo/home/CubeSats_initiative.
[Accessed: 05-Jun-2017].

[13] "THE CUBESATS OF SLS'S EM-1 - Explore Deep Space." [Online]. Available: http://exploredeepspace.com/news/the-cubesats-of-slss-em-1/. [Accessed: 05-Jun-2017].

[14] R. Mozzillo, L. Franchi, L. Feruglio, F. Stesina, and S. Corpino, "CUBESAT TEAM OF POLITECNICO DI TORINO : PAST, PRESENT AND FUTURE," in *1st Symposium on Space Educational Activities*, 2015, no. 1.

[15] "ROBUSTA - eoPortal Directory - Satellite Missions." [Online]. Available: https://directory.eoportal.org/web/eoportal/satellite-missions/r/robusta. [Accessed: 05-Jun-2017].

[16] "Nanosatellite Database." [Online]. Available: www.nanosats.eu. [Accessed: 28-Jun-2016].

[17] CalPoly, "Cubesat design specification, rev 13," *The CubeSat Program, California Polytechnic State University*. p. 42, 2014.

[18] M. Langer and J. Bouwmeester, "Reliability of CubeSats – Statistical Data, Developers' Beliefs and the Way Forward," in *AIAA/USU Conference on Small Satellites*, 2016.

[19] G. Obiols Rabasa, "Methods for dependability analysis of small satellite missions," Politecnico di Torino, 2015.

[20] S. Chien, J. Doubleday, K. Ortega, and D. Tran, "Onboard autonomy and ground operations automation for the Intelligent Payload Experiment (IPEX) CubeSat Mission," 2012.

[21] S. Chien, J. Doubleday, D. R. Thompson, and K. L. Wagstaff, "Onboard Autonomy on the Intelligent Payload EXperiment (IPEX) Cubesat Mission : A pathfinder for the proposed HyspIRI Mission Intelligent Payload Module," 2012.

[22] S. Stellmann, D. Schubert, and A. Weiss, "Historical evolution of space systems," in *60th International Astronautical Congress*, 2009, pp. 1–12.

[23] J. Naudet *et al.*, "AIM: A SMALL SATELLITE INTERPLANETARY MISSION," in *4S Symposium*, 2016.

[24] "News | JPL Selects Europa CubeSat Proposals for Study." [Online].

Available: https://www.jpl.nasa.gov/news/news.php?feature=4330. [Accessed: 06-Jun-2017].

[25] "MarCO CubeSat." [Online]. Available: http://www.jpl.nasa.gov/cubesat/missions/marco.php. [Accessed: 16-Jun-2016].

[26] "Exploration Mission 1 Secondary Payloads." [Online]. Available: https://www.nasa.gov/content/exploration-mission-1-secondary-payloads. [Accessed: 07-Jun-2017].

[27] L. McNutt, L. Johnson, P. Kahn, J. Castillo-Rogez, and A. Frick, "Near-Earth Asteroid (NEA) Scout," in *AIAA SPACE 2014 Conference and Exposition*, 2014.

[28] W. Marshall and C. Boshuizen, "Planet Labs' Remote Sensing Satellite System," *AIAA/USU Conf. Small Satell.*, 2013.

[29] "Planet Launches Satellite Constellation to Image the Whole Planet Daily." [Online]. Available: https://www.planet.com/pulse/planet-launches-satellite-constellation-to-image-the-whole-planet-daily/. [Accessed: 08-Jun-2017].

[30] "Technology | Planetary Resources." [Online]. Available: http://www.planetaryresources.com/technology/#technology-services. [Accessed: 08-Jun-2017].

[31] "Constellation of small satellites set to improve the skill of weather forecasts | Spire." [Online]. Available: https://spire.com/company/insights/news/constellation-small-satellites-set-improve-skill-w/. [Accessed: 08-Jun-2017].

[32] "Lemur-2 - Gunter's Space Page." [Online]. Available: http://space.skyrocket.de/doc_sdat/lemur-2.htm. [Accessed: 08-Jun-2017].

[33] "Home - OneWeb | OneWorld." [Online]. Available: http://oneweb.world/. [Accessed: 08-Jun-2017].

[34] L. A. Young *et al.*, "Experimental Investigation and Demonstration of Rotary-Wing Technologies for Flight in the Atmosphere of Mars," in *the 58th Annual Forum of the AHS*, 2002, no. Table 1.

[35] "KickSat -- Your personal spacecraft in space! by Zachary Manchester — Kickstarter." [Online]. Available: https://www.kickstarter.com/projects/zacinaction/kicksat-your-personal-

spacecraft-in-space. [Accessed: 08-Jun-2017].

[36] S. Hatton, *Proceedings of the 12th Reinventing Space Conference.* .

[37] J. Marshall, A. Cudmore, G. Crum, and S. Sheikh, "Big Software for SmallSats: Adapting cFS to CubeSat Missions," in *AIAA/USU Conference on Small Satellites*, 2015.

[38] C. Chouinard, R. Knight, G. Jones, and D. Tran, "An ASPEN Application : Automating Ground Operations for Orbital Express."

[39] G. Rabideau, R. Knight, S. Chien, A. Fukunaga, and A. Govindjee, "Iterative repair planning for spacecraft operations using the ASPEN system," *Int. Symp. Artif. Intell. Robot. Autom. Sp.*, vol. 440, p. 99, 1999.

[40] R. Sterritt and M. Hinchey, "Engineering Ultimate Self-Protection in Autonomic Agents for Space Exploration Missions," in *12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05)*, pp. 506–511.

[41] E. Vassev and M. Hinchey, *Autonomy requirements engineering for space missions*. Springer, 2014.

[42] W. Truszkowski *et al.*, *Autonomous and Autonomic Systems: With Applications to NASA Intelligent Spacecraft Operations and Exploration Systems*. London: Springer London, 2010.

[43] C. Rouff, "Autonomy in Future Space Missions," 2002.

[44] N. Muscettola, P. Nayak, B. Pell, and B. Williams, "The New Millennium Remote Agent: To Boldly Go Where No AI System Has Gone Before," *Artif. Intell.*, vol. 102, no. 1–2, pp. 1–39, 1998.

[45] R. Sherwood, S. Chien, and D. Tran, "Next generation autonomous operations on a current generation satellite," in *5th International Symposium on Reducing the Cost of Spacecraft Ground Systems and Operations*, 2003.

[46] European Cooperation for Space Standardization - ECSS, "ECSS-E-ST-70-11C - Space segment operability," no. July, 2008.

[47] S. Russell and P. Norvig, "Artificial Intelligence: A Modern Approach, 3rd edition," *Prentice Hall*, 2009.

[48] P. M. Frank and B. Köppen-Seliger, "New developments using AI in fault diagnosis," *Eng. Appl. Artif. Intell.*, vol. 10, no. 1, pp. 3–14, 1997.

[49] M. J. Dumskyj, S. J. Aldington, C. J. Dore, and E. M. Kohner, "The accurate assessment of changes in retinal vessel diameter using multiple frame electrocardiograph synchronised fundus photography.," *Curr. Eye Res.*, vol. 15, no. 6, pp. 625–32, Jun. 1996.

[50] Boscove, "Computer assisted vehicle service featuring signature analysis and Artificial Intelligence," 4,796,206, 1989.

[51] G. Weiss, "Multiagent systems: a modern approach to distributed artificial intelligence," no. 3. Massachusetts Institute of Technology, p. 619, 2001.

[52] Tesla, "Autopilot | Tesla." [Online]. Available: https://www.tesla.com/autopilot. [Accessed: 13-Jan-2017].

[53] Volvocars, "Autonomous driving explained | Volvo Cars," 2016. [Online]. Available: http://www.volvocars.com/intl/about/our-innovation-brands/intellisafe/autonomous-driving/this-is-autonomous-driving. [Accessed: 13-Jan-2017].

[54] Scania, "Autonomous transport systems 2016 | Scania Group." [Online]. Available: https://www.scania.com/group/en/section/pressroom/backgrounders/autonomous-transport-systems-2016/. [Accessed: 13-Jan-2017].

[55] A. M. S. Martin, S. W. Lee, and E. C. Wong, "The Development of the Msl Guidance , Navigation , and Control System for Entry , Descent , and Landing," *AAS*, pp. 529–546, 2012.

[56] C. Gulcehre, "Deep Learning - Software Links." [Online]. Available: http://deeplearning.net/software_links/. [Accessed: 28-Mar-2017].

[57] Gaisler, "LEON4." [Online]. Available: http://www.gaisler.com/index.php/products/processors/leon4. [Accessed: 22-Jun-2017].

[58] J. Brownlee, "A Tour of Machine Learning Algorithms." [Online]. Available: http://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/. [Accessed: 21-Jun-2017].

[59] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE Trans. Neural Networks*, vol. 5, no. 6, pp. 989–993, 1994.

[60] D. J. C. MacKay, "Bayesian Interpolation," *Neural Comput.*, vol. 4, no. 3, pp. 415–447, May 1992.

[61] M. F. M?ller and M. Fodslette, “A scaled conjugate gradient algorithm for fast supervised learning,” *Neural Networks*, vol. 6, no. 4, pp. 525–533, Jan. 1993.

[62] “Knowledge | Definition of Knowledge by Merriam-Webster.” [Online]. Available: https://www.merriam-webster.com/dictionary/knowledge. [Accessed: 20-Jun-2017].

[63] A. A. Hopgood, *Knowledge-Based Systems*. CRC Press, Inc, 1993.

[64] L. A. Zadeh, “The concept of a linguistic variable and its applications to approximate reasoning I,” *Inf. Sci. (Ny).*, vol. 8, no. 4, pp. 199–249, 1975.

[65] L. A. Zadeh, “Fuzzy sets,” *Inf. Control*, vol. 8, no. 3, pp. 338–353, Jun. 1965.

[66] C. C. Lee, “Fuzzy Logic in Control Systems: Fuzzy Logic Controller, Part II,” *IEEE Trans. Syst. Man. Cybern.*, vol. 20, no. 2, 1990.

[67] Y. Bai and D. Wang, *Advanced Fuzzy Logic Technologies in Industrial Applications*. Springer, 2006.

[68] H. T. Nguyen, N. R. Prasad, C. L. Walker, and E. a Walker, *A First Course in Fuzzy and Neural Control*. 2003.

[69] S. Luke *et al.*, *Essentials of Metaheuristics Second Edition*. 2015.

[70] C. W. Ahn, “Practical genetic algorithms,” *Stud. Comput. Intell.*, vol. 18, pp. 7–22, 2006.

[71] J. Figueira, S. Greco, and M. Ehrgott, *Multiple Criteria Decision Analysis: state of the art surveys*. Springer, 2005.

[72] P. S. Oliveto, J. He, and X. Yao, “Time Complexity of Evolutionary Algorithms for Combinatorial Optimization: A Decade of Results,” *Int. J. Autom. Comput.*, vol. 4, no. 3, pp. 281–293, 2007.

[73] T. W. Manikas and J. T. Cain, “Genetic Algorithms vs . Simulated Annealing : A Comparison of Approaches for Solving the Circuit Partitioning Problem,” 1996.

[74] C. W. Ahn, *Advances in evolutionary algorithms : theory, design and practice*. Springer, 2006.

[75] R. Rojas, “Genetic Algorithms,” *Neural Networks*, pp. 429–450, 1996.

[76] T. Back, “Selective pressure in evolutionary algorithms: a characterization

of selection mechanisms," in *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, pp. 57–62.

[77] D. Pierce and A. Petro, "NASA Perspectives on Cubesat Technology and Highlighted Activities," 2016.

[78] Jam Woerner, "ESA COUNCIL AT MINISTERIAL LEVEL 2016: SUCCESS, TINGED WITH A BIT OF DISAPPOINTMENT," *Journal of Fusion Energy*, 04-Dec-2016. [Online]. Available: http://link.springer.com/10.1007/s10894-015-0034-1. [Accessed: 16-Jun-2017].

[79] R. Walker, D. Koschny, and C. Bramanti, "Miniaturised Asteroid Remote Geophysical Observer (M-ARGO): a stand-alone deep space CubeSat system for low- cost science and exploration missions," 2017.

[80] J. P. Cohen, H. Z. Lo, T. Lu, and W. Ding, "Crater Detection via Convolutional Neural Networks," in *47th Lunar and Planetary Science Conference*, 2016.

[81] S. Singh and M. Singh, *Progress in pattern recognition*. Springer, 2007.

[82] A. Criminisi, "Decision Forests: A Unified Framework for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning," *Found. Trends® Comput. Graph. Vis.*, vol. 7, no. 2–3, pp. 81–227, 2011.

[83] S. Chien *et al.*, "Onboard Autonomy on the Intelligent Payload EXperiment CubeSat Mission," *J. Aerosp. Inf. Syst.*, no. March, pp. 1–9, 2016.

[84] C. Stauffer and W. Grimson, "Learning Patterns of Activity Using Real-Time Tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 747–757, 2000.

[85] C. Koch and S. Ullman, "Shifts in Selective Visual Attention: Towards the Underlying Neural Circuitry," in *Matters of Intelligence*, Dordrecht: Springer Netherlands, 1987, pp. 115–141.

[86] J. F. Martins, V. F. Pires, and A. J. Pires, "Unsupervised neural-network-based algorithm for an on-line diagnosis of three-phase induction motor stator fault," *IEEE Trans. Ind. Electron.*, vol. 54, no. 1, pp. 259–264, 2007.

[87] W. Lafayette, C. Clifton, C. Sciences, and W. Lafeyette, "CERIAS Tech Report 2003-45 Change Detection in Overhead Imagery Using Neural

Networks by Christopher Clifton Information Assurance and Security," 2003.

[88] L. Feruglio and S. Corpino, "Neural networks to increase the autonomy of interplanetary nanosatellite missions," *Rob. Auton. Syst.*, vol. 93, pp. 52–60, 2017.

[89] P. Michel, A. Cheng, M. Küppers, and P. Pravec, "Science case for the Asteroid Impact Mission (AIM): A component of the Asteroid Impact & Deflection Assessment (AIDA) mission," *Adv. Sp. Res.*, vol. 57, pp. 2529–2547, 2016.

[90] F. Nimmo, J. R. Spencer, R. T. Pappalardo, and M. E. Mullen, "Shear heating as the origin of the plumes and heat flux on Enceladus," *Nature*, vol. 447, no. 7142, pp. 289–291, May 2007.

[91] M. F. A'Hearn, M. J. S. Belton, W. A. Delamere, and J. Kissel, "Deep Impact: excavating comet Tempel 1.," *Science*, vol. 310, no. 5746, pp. 258–64, Oct. 2005.

[92] A. A. Hopgood, *Intelligent systems for engineers and scientists*. CRC Press, 2012.

[93] K. Lee, "Theoretical study of information capacity of Hopfield neural network and its application to expert database system," Iowa State University, 1991.

[94] ESA, "ASTEROID IMPACT MISSION: DIDYMOS REFERENCE MODEL," 2014.

[95] J. R. Shewchuk, "An Introduction to the Conjugate Gradient Method Without the Agonizing Pain," 1994.

[96] L. Franchi, L. Feruglio, R. Mozzillo, and S. Corpino, "Model predictive and reallocation problem for CubeSat fault recovery and attitude control," *Mech. Syst. Signal Process.*, vol. 98, pp. 1034–1055, 2018.

[97] R. de Neufville, "Measurement of Utility," *Appl. Syst. Anal. Eng. Plan. Technol. Manag.*, 1990.

[98] R. de Neufville, "Multiattribute Utility," *Appl. Syst. Anal. Eng. Plan. Technol. Manag.*, 1990.

[99] A. M. Ross and D. E. Hastings, "The Tradespace Exploration Paradigm," *INCOSE Int. Symp.*, vol. 15, no. 1, pp. 1706–1718, 2005.

[100] B. A. Corbin, “The Value Proposition of Distributed Satellite Systems for Space Science Missions,” Massachusetts Institute of Technology, 2015.

[101] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by Simulated Annealing,” *Sci. New Ser.*, vol. 220, no. 4598, pp. 671–680, 1983.

[102] I. Das and J. E. Dennis, “Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems,” *SIAM J. Optim.*, vol. 8, no. 3, pp. 631–657, Aug. 1998.

[103] D. M. Olsson and L. S. Nelson, “The Nelder-Mead Simplex Procedure for Function Minimization,” *Technometrics*, vol. 17, no. 1, p. 45, Feb. 1975.

[104] R. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory,” *Proc. Sixth Int. Symp. Micro Mach. Hum. Sci.*, pp. 39–43, 1995.

[105] J. D. Schaffer, “Multiple objective optimization with vector evaluated genetic algorithms,” *1st Int. Conf. Genet. Algorithms*, no. JANUARY 1985, pp. 93–100, 1985.

[106] P. Abell *et al.*, “Asteroid Impact & Deflection Assessment (Aida) Mission,” no. May, 2012.

[107] E. Riddle, “Use of optimization methods in small satellite systems analysis,” *Proc. AIAA/USU Conf. Small Satell.*, pp. 1–8, 1998.

[108] Gwo-Ching Liao and Ta-Peng Tsao, “Application of a fuzzy neural network combined with a chaos genetic algorithm and simulated annealing to short-term load forecasting,” *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 330–340, 2006.

[109] B. Corbin and T. Steiner, “Multidisciplinary System Design Optimization for a Distributed Solar Observation Constellation!,” 2014.

[110] J. Sobieszczanski-Sobieski, A. Morris, and M. van Tooren, *Multidisciplinary Design Optimization Supported by Knowledge Based Engineering*. 2015.

[111] N. Srinivas and K. Deb, “Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms,” *Evol. Comput.*, vol. 2, no. 3, pp. 221–248, Sep. 1994.

[112] A. Jafarsalehi, P. M. Zadeh, and M. Mirshams, “Collaborative Optimization of Remote Sensing Small Satellite Mission using Genetic Algorithms,” *Trans. Mech. Eng.*, vol. 36, no. 2, pp. 117–128, 2012.

[113] S. Gotshall and B. Rylander, “Optimal population size and the genetic algorithm,” *Proc. Genet. Evol. Comput. Conf.*, pp. 1–5, 2000.

Appendix A – Interesting images acquired through the research

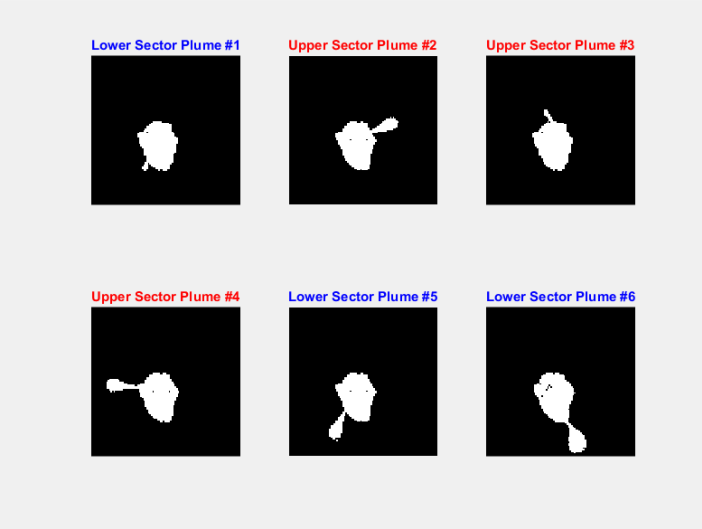


Figure 67 Plume events: detection of upper or lower direction

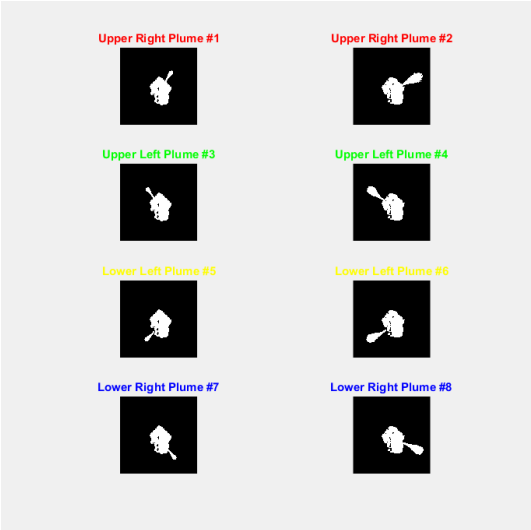


Figure 68 Plume events: detection of four directions

Figure 69 Plume events: detection of eight directions

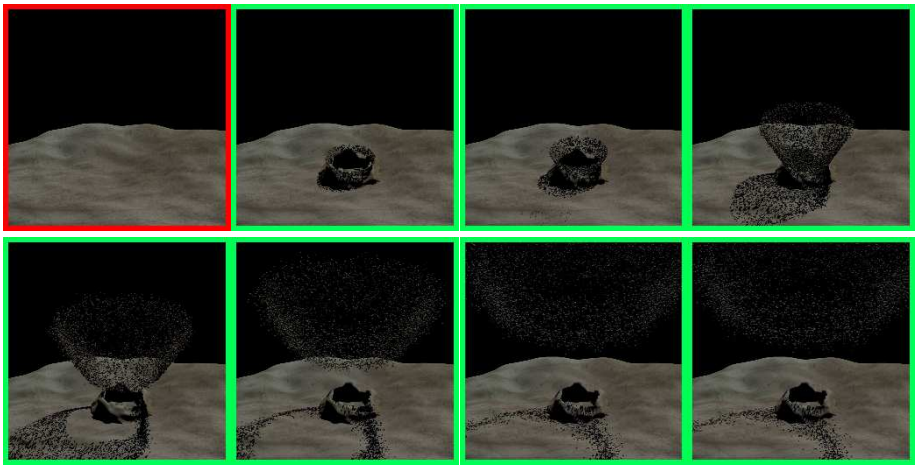


Figure 70 Impact sequence on an asteroid, simulation with dark sky in the background

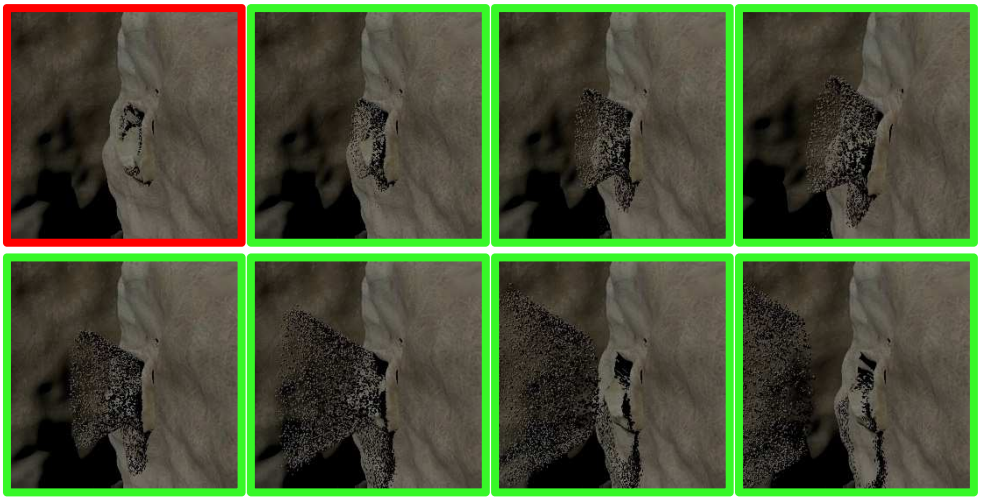


Figure 71 Impact sequence on an asteroid, simulation with main body in the background

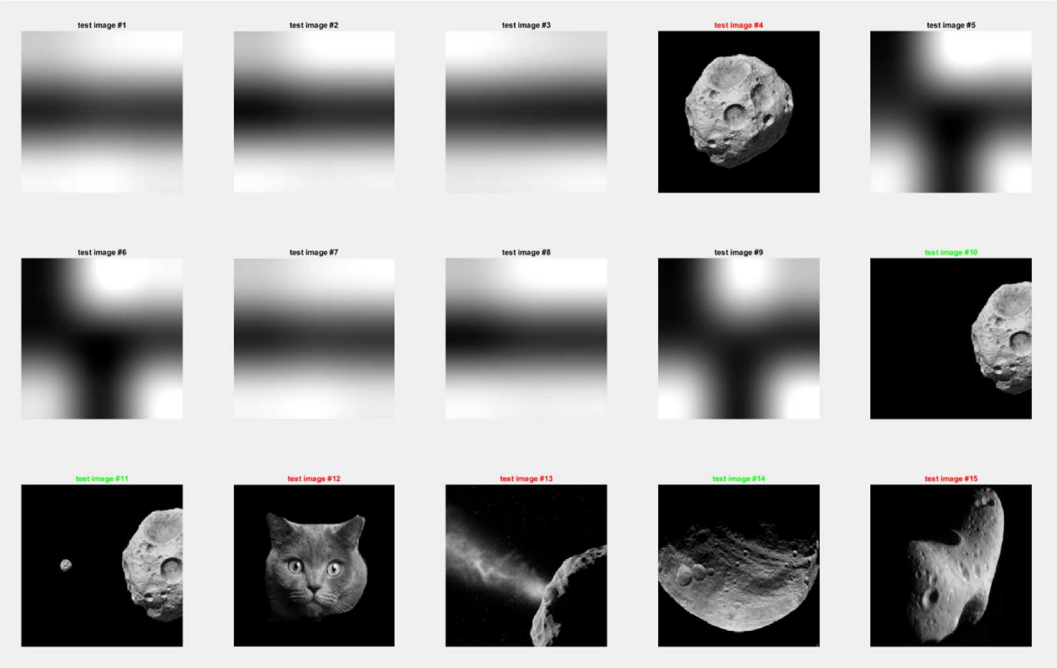


Figure 72 Early experimentations with Neural Networks: cats are recognized as fully pictured asteroid. The picture right from the cat is wrongly classified.

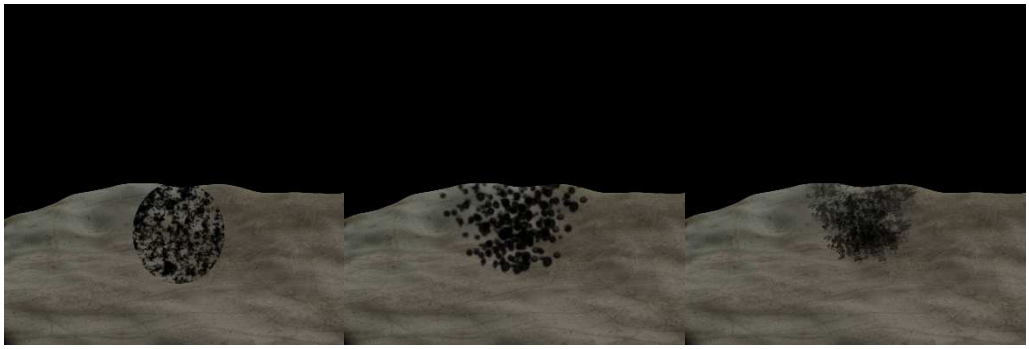


Figure 73 Experimenting with the overlay training methodology described in the thesis

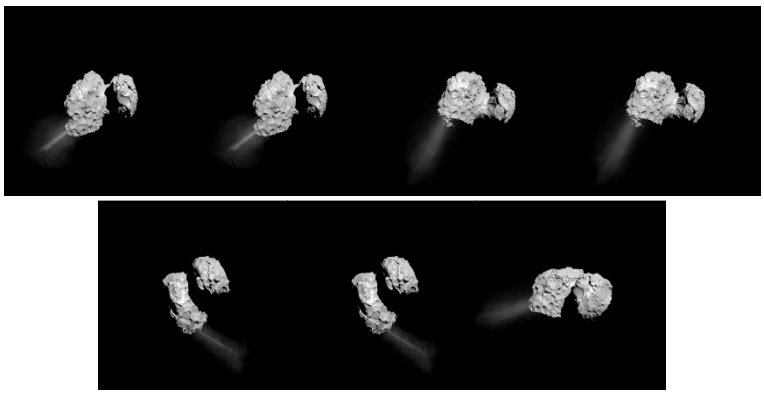


Figure 74 67P plume events as modelled on blender®

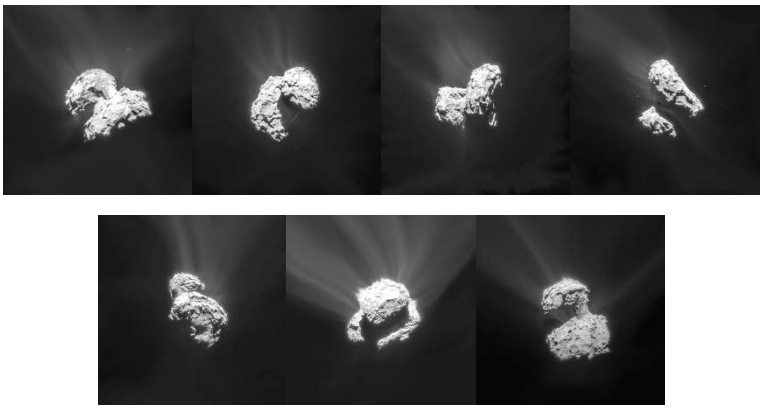


Figure 75 67P plume events as photographed by the Rosetta mission

Appendix B - Asteroid modelling on blender®

The first operation performed was the creation of a cube, from the submenu "create".

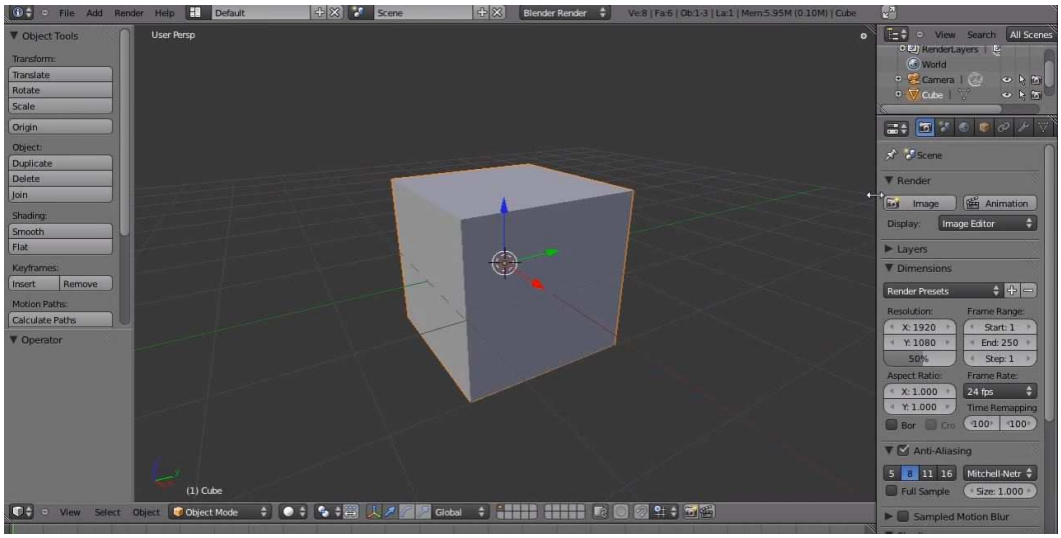


Figure 76 Asteroid Modelling: creation of the starting cube

In Blender, each object has its own reference axes, so there is no need to create a special coordinate system.

The next step is to add the "Modifiers" from the corresponding submenu. After selecting "Add Modifiers" the first of them will be "Subdivision Surface". Soon after, under the heading "Subdivision", the values "View" and "Render" will be brought to the upper limit, i.e. six. The result of this operation is shown in Figure 76.

Figure 77 Asteroid Modelling: Subdivision Surface Modifier

Figure 78 Asteroid modelling: texture

Next, the "Smooth" button under "Shading" in the left submenu "Tools" has been selected.

The second modifier added, always in the same way, is "Displacement". It allows the introduction of the ripples on the surface in question, according to a user-determined texture, using the “Add Texture” command. The selected texture is shown in Figure 78. It is possible to choose additional parameters to customize in the sub-menu "Texture". To create the more precise geometries, selecting "subsurface" with the right mouse button individual faces of the intermediate solid can be selected and, using Tab and G keys, deformed at will. To have a greater level of detail, it is advisable to add another level of subsurface, setting the value of "View" and "Render" on two. Eventually, the asteroid was put into rotation around its axis through a 500 frames animation.

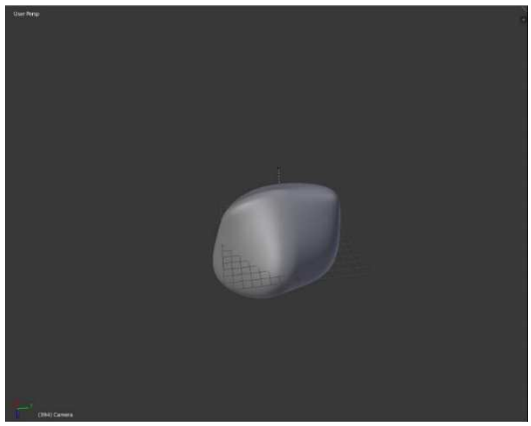


Figure 79 Asteroid modelling: editing the geometry

Figure 80 Asteroid modelling: final result

The plume can be added by creating a reference plane on which is to be placed the source point. This done, the values shown in Figure 81 were inserted in the "Particle" submenu.

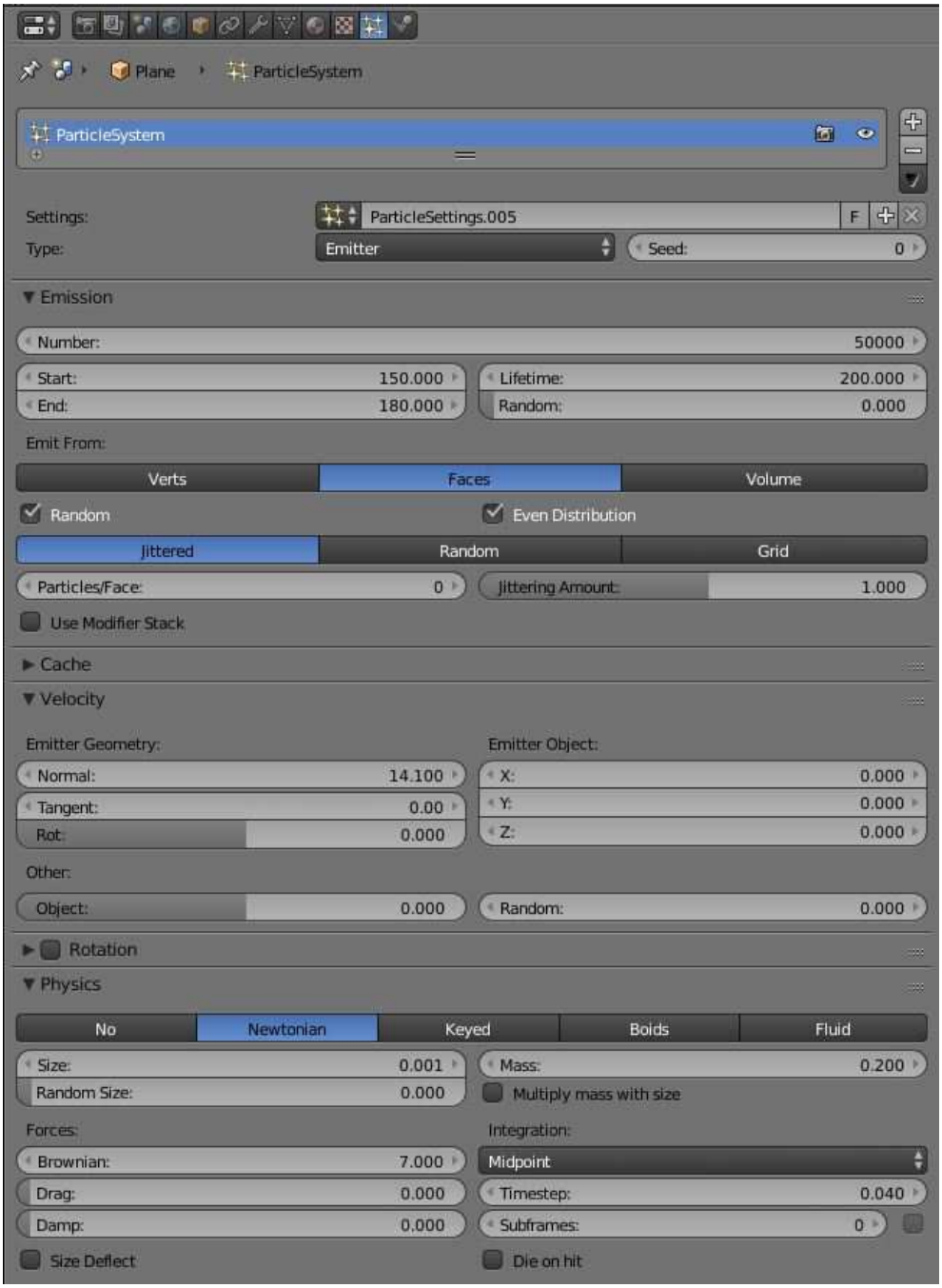


Figure 81 Plume modelling parameters

On completion of modelling, we can add a light via the "Lamp" submenu.

The emission of the plume starts at frame 150 and ends at 180 (despite the emitted particles continue to be still visible for 200 frames).