POLITECNICO
MILANO 1863

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska Curie grant agreement No 813644.

Politecnico di Milano, Milan, Italy
Department of Aerospace Science and Technology
Doctoral programme in Aerospace Engineering

Data-Driven Image Processing for Enhanced Vision-Based Applications Around Small Bodies with Machine Learning

PhD dissertation of:
Mattia Pugliatti

Supervisor:
Prof. Francesco Topputo, PhD

Tutor:
Prof. Aldo Frezzotti, PhD

Coordinator:
Prof. Pierangelo Masarati, PhD

Mattia Pugliatti: *Data-Driven Image Processing for Enhanced Vision-Based Applications Around Small Bodies with Machine Learning*

Supervisor:
Prof. Francesco Topputo, PhD

Location:
Milan, Italy

*"One must still have chaos in oneself
to be able to give birth to a dancing star"*

Friedrich Wilhelm Nietzsche

Copyright © 2023, Mattia Pugliatti

All Rights Reserved

Abstract

The space sector is undergoing rapid growth, especially in near-Earth orbits, promising unprecedented benefits through integrated space-based services. At the same time, CubeSats are reshaping deep space by diversifying scientific objectives and complementing traditional missions. Profiting from this favorable environment, a surge in deep-space missions dedicated to the exploration and exploitation of the Solar System is on the horizon. Within this context, small celestial bodies, such as asteroids and comets, emerge as intriguing targets due to their abundance, proximity to Earth's orbit, ancient origins, importance for planetary defense, potential for resource utilization, and the quest for extraterrestrial life. However, the operation of a large fleet of spacecraft exploring these deep space bodies poses critical challenges when approached with the current ground-based paradigm. Driven by the need for real-time decision-making and cost-effective solutions, technological advancements are gearing towards autonomous spacecraft operations. Within this context, artificial intelligence enhancements on computer vision tasks are posed to enrich perception and spatial comprehension of the surrounding environment, enabling intelligent spacecraft to operate effortlessly and autonomously. Image segmentation and visual-based navigation, in particular, are investigated in this manuscript using neural networks and machine learning approaches. Data-driven image processing options are also assessed for the future CubeSat mission Milani, which will visit the Didymos binary system. Milani's semi-autonomous vision-based capabilities pave the way for adopting data-driven algorithms in deep space. Assessing the performance of these techniques is as important as highlighting their drawbacks and the challenges associated with their development, primarily related to the availability of high-quality training data. The integration of artificial intelligence and autonomous capabilities holds the potential to revolutionize our engagement with minor bodies, shaping the future of space exploration.

Sommario

Il settore spaziale è in rapida crescita, soprattutto nelle basse orbite terrestri, promettendo vantaggi senza precedenti attraverso servizi integrati. Allo stesso tempo, i piccoli satelliti stanno rivoluzionando il nostro modo di esplorare lo spazio profondo, diversificando gli obiettivi scientifici e complementando missioni tradizionali. Approfittando di questo ambiente favorevole, si prospetta un'ondata di missioni dedicate all'esplorazione e allo sfruttamento del sistema solare. In questo contesto, piccoli corpi celesti, come asteroidi e comete, emergono come intriganti obiettivi per la loro abbondanza, vicinanza all'orbita terrestre, antiche origini, l'importanza per la difesa planetaria, potenziale sfruttamento delle risorse e la ricerca di vita extraterrestre. Tuttavia, la gestione di una grande flotta di veicoli spaziali pone sfide critiche se affrontate con l'attuale paradigma basato sulle operazioni da terra. Spinti dalla necessità di un processo decisionale in tempo reale e di soluzioni economicamente vantaggiose, i progressi tecnologici si stanno orientando verso operazioni autonome dei veicoli spaziali. In questo contesto, i miglioramenti dell'intelligenza artificiale nell'analisi delle immagini sono proposti per arricchire la percezione e la comprensione dell'ambiente circostante, consentendo a dei satelliti intelligenti di operare senza sforzo e in modo autonomo. In particolare, la segmentazione delle immagini e la navigazione ottica, sono state investigate in questo manoscritto utilizzando reti neurali e approcci di apprendimento automatico. Algoritmi basati sui dati sono valutati anche per la futura missione Milani, un piccolo satellite che visiterà il sistema binario Didymos. Le capacità semi-autonome di Milani aprono la strada all'adozione di algoritmi basati sui dati nello spazio profondo. Valutare le prestazioni di queste tecniche è importante quanto evidenziare i loro svantaggi e le sfide associate al loro sviluppo, principalmente legate alla disponibilità di dati di addestramento di alta qualità. L'integrazione tra intelligenza artificiale e capacità autonome ha il potenziale di rivoluzionare il modo in cui interagiamo con i corpi minori, plasmando il futuro dell'esplorazione spaziale.

Acknowledgments

This has been a hell of a ride. Luckily, it has not been a solitary one. We are the sum of the people we meet, and I feel lucky to have met all of you across such a strange and complex journey. Allow me to express my heartfelt appreciation to each of you.

First and foremost, I thank my supervisor and mentor. Francesco, you have taught me many lessons I will treasure for the rest of my life, but above all, you have shown me what it means to perform research in a passionate, dedicated, and professional way.

Gianmario, we have been in this together since day one. This adventure would not have been the same without your support, encouragement, and friendship.

Michele, thanks for all the patience, the talks, and the unwavering support you have always dispensed. I wish you the utmost success because you deserve every bit of it.

I have spent long enough at the DART lab to see it evolving to become what it is today. It is not easy to find such a dedicated group of people working together. Thanks for making these years lighter, more enjoyable, and endlessly entertaining. In particular, I would like to extend my gratitude to Eleonora, Gianfranco, Carminozzo, Alessandra, Felice, Antonio, Paolo, and Vittorio.

A special acknowledgment to Carmine and Andrea. Carmine for being the perfect companion in our Blender (mis)-adventures. As you always like to say, you have evolved from a M.Sc. student to a colleague to a friend, and I am delighted it happened. Andrea, for always being there and willing to discuss everything, always dispensing great pieces of advice just when they are needed the most. For both of you, your friendship and support, both in and out of work, have been truly indispensable. I will deeply miss our conference moments.

I would also like to spend a few words of gratitude towards other fellow researchers from the DAER: Marco, Alessandro, Federico, Giovanni, and Alessio. You made our lunch and coffee breaks in the department much more enjoyable experiences, especially during the harsh times of COVID.

Thanks to Roberto Furfaro for hosting me for four enlightening months at the University of Arizona and the rest of the Arizona gang: Andrea, Mario, Enrico, Laura, Marco, Luca, and many others.

Thanks also to Marko Jankovic for hosting me for three months at the DFKI and thanks to all the new and old friends that made this Bremen experience so unique: Eleonora, Gianfranco, Claudio, Adriana, Stefano, Hans, and most importantly, Shubham. Thanks for showing me another way of living and doing research. Thanks for initiating me to bouldering and for being such a great friend. I am looking forward to our stand-up comedy show soon.

Thanks to all the fellow friends of Stardust-R. You made it a delightful experience and demonstrated the power and difficulties of collaborative research on a large scale.

Aurelio, in an ideal academic world, we would be rivals due to our similar research topics. In the real world, I cherish the friendship and collaboration we have built.

I also would like to thank Chiara and her family. We shared this path halfway through. It was complicated and riddled with unforeseen challenges. Your support signified a lot over the years and meant the world to me. Even though we are now walking separate paths, I wanted to thank you for the patience and love you have dispensed.

The Milanese task force of historical friends: Stefano T., Luca, and Viviana. Thanks for successfully managing over so many years to always make me feel at home whenever I am back in Milano.

Stefano B., for being such a vital source of support and inspiration to look at, both professionally and personally, since so many years.

Carla, you entered my life unexpectedly and significantly impacted it when I needed it the most. Every blessing ignored becomes a curse, and you truly are a blessing. Your positive influence and constant optimism have been nothing short of transformative. Thank you for having taken the risk and having put an Italian in your life.

This manuscript has been written as a digital nomad while traveling worldwide. Thanks to those who have encouraged me to do this physical and spiritual journey, not thinking I had become crazy. In particular, thanks to Stefano and Carla, who have hosted and supported me during such a complex time, but also all the amazing people from the Lost Creator House in Bali and all the Balinese people, who have some of the most incredible smiles on the planet.

Finally, a profound thanks to my family. You have always put my education first and taught me the value of hard work, a strong ethic, and optimism, with a pinch of belief in the power of the universe. Even though you stopped to understand what I do (and why) many years ago, knowing I have your unwavering support is a powerful source of encouragement.

Grazie

Mattia Pugliatti
Milan, November 2023

Table of contents

1

Introduction

"Considerate la vostra semenza:
fatti non foste a viver come bruti,
ma per seguir virtute e canoscenza"

"Consider your origins:
you were not made to live as brutes,
but to follow virtue and knowledge"

Dante Alighieri, Inferno XXVI

The space sector is experiencing flourishing growth, and integrated, space-based services will soon benefit humanity at unprecedented levels. The momentum characterizing the near-Earth space will also benefit outer space. Evidence is mounting that the near future will be characterized by a large amount of deep-space missions dedicated to the exploration and exploitation of the Solar System [1–7].

This expansion will be fueled by CubeSats, modular miniaturized spacecraft made of several units, which are already revolutionizing the way current exploration of the Solar System is made by diversifying and complementing the scientific objectives of larger missions [1, 2]. Currently, most miniaturized spacecraft have thus far been deployed into near-Earth orbits. However, a multitude of interplanetary CubeSats will soon be employed for deep-space missions as well [2–4, 6, 7]. The operation of a large fleet of spacecraft in deep space, however, challenges the current assumptions for ground-based operations. This, coupled with limitations during critical operations related to two-way communication delays and the push to reduce costs, boosts the adoption of autonomous technologies.

In this context, small bodies, such as asteroids and comets, represent an exciting playground for various reasons. Bodies such as near-Earth asteroid (NEA) are characterized by orbital parameters close to those of Earth, making them accessible targets even with low-cost and small platforms. As remnants of the ancient building blocks from which our planets and other celestial objects formed, these bodies provide insights into the origin of our cosmic neighborhood. Moreover, their study offers invaluable information regarding potential threats to our planet, the prospect for extraction and utilization of extraterrestrial resources, and even the intriguing possibility of life beyond Earth.

1.1 Context

In 1772, the German astronomer Johann Elert Bode published a formula, referred to as the Titius-Bode law, that appeared to predict the orbits of all the known planets (at the time Mercury, Venus, Earth, Mars, Jupiter, and Saturn) while also anticipating the presence of a planet yet to be discovered between the orbits of Mars and Jupiter [8]. The formula gained credibility in 1781 after the discovery of Uranus by William Herschel in a position close to the predicted one. It also motivated the formation in 1800 of a group of experienced astronomers led by the editor of the German astronomical journal "journal Monatliche" (Monthly Correspondence) to search for the missing planet. This group referred to itself as the "celestial police" [8] and included the astronomer Giuseppe Piazzi, a catholic priest at the Academy of Palermo, Sicily, who already on the 1st of January 1801 (before actually getting invited to join), discovered a "star-like moving object". Initially considered a comet, the object exhibited bizarre behavior, moving slower than expected and with a relatively uniform motion. In Piazzi's words: "...it has occurred to me several times that it might be something better than a comet" [8].

Piazzi published the discovery in the September issue of the Monthly Correspondence in 1801, naming the body "Ceres Ferdinandea"; Ceres in honor of the Roman goddess of agriculture, whose oldest temple was in Sicily, and Ferdinandea in honor of Piazzi's monarch and patron, King Ferdinand III of Sicily (the latter name not being accepted by the scientific community). However, at the time of publication, Ceres was too close to the Sun's glare for other astronomers to observe and would have become visible again towards the end of the year, lost in an unpredictable position in the sky. Interested in this challenge, the mathematician Carl Friedrich Gauss specifically developed an efficient orbit determination method to predict Ceres' reappearance in the sky for everyone to observe [8]. Based on Gauss's solution, Ceres was thus observed again at the end of 1801 in the expected position, and the scientific community verified its discovery, proving the existence of the missing planet.

As astronomers rushed to observe the missing planet, to much of their surprise, they ended up discovering other similar bodies. It was then clear that a single missing planet did not exist but rather a population of smaller and irregular objects believed to be chunks of the original missing planet. These bodies were observed from telescopes of the time to be similar to stars, yet distinguishable from them due to their apparent motions, and similar to comets yet lacking typical cometary features. To distinguish them from stars, planets, and comets, William Herschel thus proposed to name them "asteroids", meaning "star-like, star-shaped" in Greek [8].

With the realization of the existence of a population of these bodies, our comprehension of the Solar System changed forever, transforming it into a vibrant and complex dynamical system. In addition to the eight known planets, countless smaller bodies orbit the Sun. From dust grains and small rocks with little gravity to dwarf spherical-like planetoids, their population is estimated in the millions, with new bodies being constantly discovered by survey missions, even beyond the Solar System. Nowadays, bodies such as asteroids, comets, and dwarf planets are usually classified together as minor bodies or small bodies [9] to differentiate them from planets. This nomenclature will be used throughout the rest of this manuscript.

While comets have been observed for millennia with the naked eye (especially when releasing gas and dust in the proximity of the Sun) and have been depicted in the arts and folklore since generations [10], the history of asteroids observation has just started since most of them are faint or small and are visible only for a short amount of time, detectable only with the use of proper instrumentation.

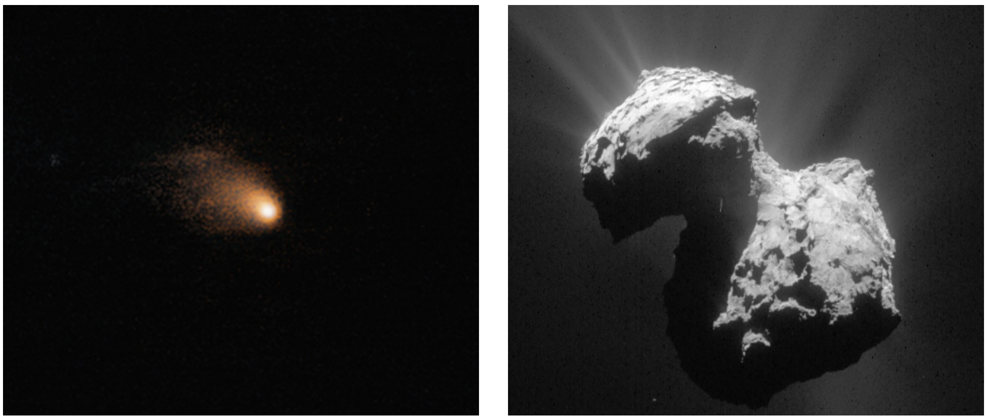

(a) 67P seen from Earth. **(b)** 67P seen from the Rosetta spacecraft.

Figure 1.1: Image of the comet 67P/Churyumov–Gerasimenko seen from the Very Large Telescope in 2014 (a) and from the Rosetta spacecraft in 2015 (b). Credits: ESA.

Given the significant constraints posed by Earth-based and space-based telescopes, the investigation of these bodies in close proximity has remained a crucial endeavor for space exploration. As technological advancements have made this approach feasible, robotic spacecraft have been employed to approach these bodies, utilizing either flyby or rendezvous strategies.

A flyby refers to a mission or trajectory in which the spacecraft passes close to a celestial body or another object, typically at a high relative speed, without entering orbit around it. During the flyby, the spacecraft's instruments and sensors can capture valuable data, images, and measurements as it passes by the target. This data can include information about the body's surface features, composition, atmosphere, magnetic field, and others. Since establishing a long-term orbit around the body is challenging due to extreme distances, harsh radiation environments, operational costs, and high velocities, flybys are a reasonable and cheap alternative to orbiting the body. For this reason, they are the most accessible form of exploration.

However, because the encounter with the body is bounded in time and space, the amount and quality of scientific data are severely limited to fully characterize the body.

A rendezvous refers to a mission or trajectory planned and controlled to keep a spacecraft around the target body. Because the rendezvous is not limited to a short amount of time and space, it can significantly enhance the scientific data that can be generated. This comes at the cost of the complexity and robustness of the spacecraft, overall mission, and trajectories adopted. In a rendezvous, it is essential to ensure the spacecraft does not impact on the asteroids as well as that the spacecraft performs as expected for the entire duration of the mission, with a considerable impact on the operational costs compared to a flyby.

The history of exploration of minor bodies started roughly *50* years ago, with an incidental flyby with an unknown asteroid by the Pioneer 10 spacecraft in *1972*. From that moment onward, thanks to various flyby missions first and rendezvous missions later, a fair number of bodies have been visited by robotic spacecraft. These are summarized in Table 1.1 and Table 1.2, divided respectively into flyby and rendezvous missions. The data about flyby missions has been collected from various sources [1], [2], [3], [9, 11], while the one about rendezvous missions are specified in Table 1.2. Landers and other tools/instruments/sub-units of the main spacecraft have been omitted from the tables for clarity, as well as the numerous projects, proposed, and planned missions.

Impact missions such as DART and Deep Impact have been designated as flyby missions for clarity. Also, note that in DART, the mothercraft impacted asteroid Dimorphos (the secondary body of the 65803 Didymos binary system), while Deep Impact used an impactor unit that crashed against the comet 9P/Tempel.

Up to September 2023, a total of *37* missions have been successfully executed using *27* different platforms. The total number of different bodies visited within the Solar System is *30*, comprised of *7* comets visited with a flyby or impact mission, *16* asteroids visited with a flyby or impact mission, and *7* bodies (*5* asteroids, *1* dwarf planet, and *1* comet) visited by a rendezvousing spacecraft. In Figure 1.2, it is possible to visualize the data from Table 1.1 and Table 1.2 displayed as a cumulative number of successful missions dedicated to minor bodies exploration since *1972*.

A mosaic view of most of the minor bodies visited so far is illustrated in Figure 1.3 from [4], illustrating a variety of irregular shapes and a rich presence of surface features such as ridges, craters, and boulders.

[1] https://www.planetary.org/space-missions/every-small-worlds-mission, last accessed 8th of August, 2023.

[2] https://en.wikipedia.org/wiki/List_of_missions_to_minor_planets, last accessed 8th of August, 2023.

[3] https://en.wikipedia.org/wiki/List_of_minor_planets_and_comets_visited_by_spacecraft, last accessed 8th of August, 2023.

[4] https://www.planetary.org/space-images/asteroids-and-comets-visited-by-spacecraft, last accessed 8th of August, 2023.

Table 1.1: Previous flyby missions to asteroids and dwarf planets (first half of the table) and comets (second half of the table) in chronological order of Closest Approach (CA).

Mission	Year	Body's name	CA [km]	Notes
Pioneer 10	1972	Unnamed	$8.85 \cdot 10^6$	Incidental
Pioneer 10	1972	307 Nike	$8.80 \cdot 10^6$	Incidental
Galileo	1991	951 Gaspra	*1604*	Incidental
Galileo	1993	243 Ida	*2410*	Incidental
Galileo	1993	243 Dactyl	*2410*	Incidental
NEAR	1997	253 Mathilde	*1212*	
Deep Space 1	1999	9969 Braille	*28.3*	
Cassini–Huygens	2000	2685 Masursky	$1.50 \cdot 10^6$	Incidental
Stardust	2002	5535 Annefrank	*3079*	
New Horizons	2006	132524 APL	$1.02 \cdot 10^5$	Incidental
Rosetta	2008	2867 Šteins	*800*	
Rosetta	2010	21 Lutetia	*3162*	
Chang'e-2	2012	4179 Toutatis	*3.2*	
New Horizons	2015	134340 Pluto	*12500*	
New Horizons	2019	486958 Arrokoth	*3538*	
DART	2022	65803 Didymos	*0*	Impact
LICIACube	2022	65803 Didymos	*56.7*	
ICE	1985	21P/Giacobini-Zinner	*7800*	
Vega 1	1986	1P/Halley	*8889*	
Vega 2	1986	1P/Halley	*8030*	
Suisei	1986	1P/Halley	$1.51 \cdot 10^5$	
Sakigake	1986	1P/Halley	$6.99 \cdot 10^6$	
Giotto	1986	1P/Halley	*596*	
ICE	1986	1P/Halley	$31.00 \cdot 10^6$	
Giotto	1992	26P/Grigg–Skjellerup	*200*	
Deep Space 1	2001	19P/Borrelly	*2171*	
Stardust	2004	81P/Wild	*240*	
Deep Impact	2005	9P/Tempel	*500*	Impactor unit
EPOXI	2010	103P/Hartley	*700*	
Stardust	2011	9P/Tempel	*181*	

Nowadays, the exploration of minor bodies is propelled by one or a combination of the following motivations [1, 9]: scientific, planetary defense, technology demonstration, and resource exploitation.

From a scientific perspective, these bodies are thought to enclose valuable information on the primordial state of the Solar System. As remnants of the primordial Solar System, they enact a living library of our cosmic history and are

Table 1.2: Previous rendezvous missions towards minor bodies in chronological order of arrival.

Mission	Year	Body's name	Reference
NEAR	2000	433 Eros	[12]
Hayabusa-I	2005	25143 Itokawa	[13]
Dawn	2011	4 Vesta	[14]
Rosetta	2014	67P/Churyumov–Gerasimenko	[15]
Dawn	2015	1 Ceres	[14]
Hayabusa-II	2018	162173 Ryugu	[16]
OSIRIS-REx	2018	101955 Bennu	[17]

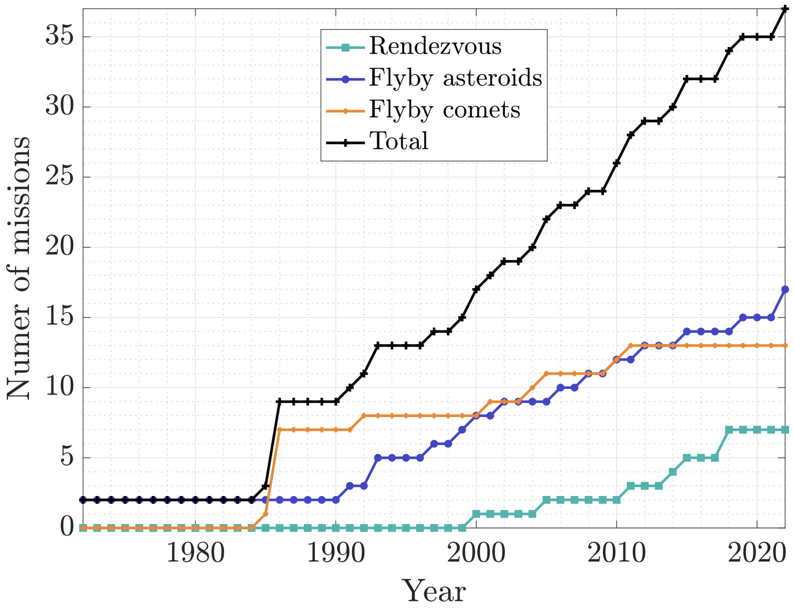


Figure 1.2: Cumulative number of missions towards minor bodies per year since *1972*, divided into asteroid and dwarf planet flybys, comet flybys, rendezvous, and total. Considering only the data from *1985* onward, the number of missions per year exhibits a linear trend. A first-order degree fit of the data generates the coefficients $p_1 = 0.7709(0.7326, 0.8092)$, $p2 = -1523(-1600, -1447)$ and 95% confidence bounds, achieving a Root Mean Squared Error (RMSE) equal to 2.0033 and assuming the cumulative number of succesf-ful missions expressed as $p_1 \cdot Year + p_2$.

thus fundamental to understanding how planetary systems form [9]. Their extensive presence poses both a threat to our planet and, at the same time, offers plenty of opportunities. For example, as they are cheaper to reach, minor bodies present an exciting opportunity for technology demonstration missions; they can be used as sandboxs to boost the progress of advanced technologies. From a resource

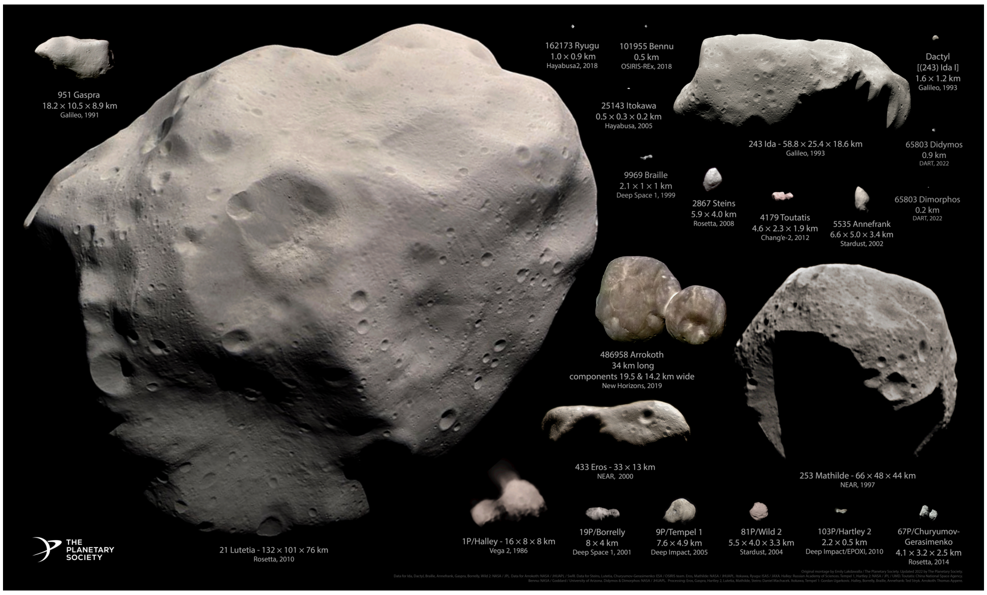


Figure 1.3: Visited minor bodies up to September 2022 (excluding Pluto, Ceres, and Vesta). Original montage by Emily Lakdawalla, The Planetary Society.

exploitation point of view, minor bodies could represent the frontier of a new gold race for rare metals extraction and additive manufacturing in space [1]. Finally, from a planetary defense perspective, minor bodies pose a real threat to activities on Earth and in space. Many techniques proposed to deflect asteroids and comets have only been theorized or tested in scaled laboratory environments. Only three missions have ever demonstrated and tested the effectiveness of a deflection technique, all through the usage of a kinetic impactor: Deep Impact [18] on 9P/Tempel, Hayabusa-II [16] on Ryugu with small-scale impactor experiments, and more recently DART [19] on Dimorphos. Advantageously, as the potential destructiveness of a minor body is proportional to its size, it also makes it easier to observe the most dangerous ones in time to prepare a deflection strategy accordingly (if any is needed) [20]. Figure 1.4 is an infographic [5] displaying population, energy, and impact frequencies as a function of the size of the known and unknown population of asteroids.

1.2 Motivation

The current paradigm for exploring small bodies mirrors for most of the operations an established strategy for interplanetary missions: with open-loop control and meticulous oversight from the ground, the spacecraft is continuously operated from a dedicated control center for months or years.

[5] https://www.esa.int/Space_Safety/A_burst_of_asteroid_activity_in_Europe, last accessed 21st August, 2023.

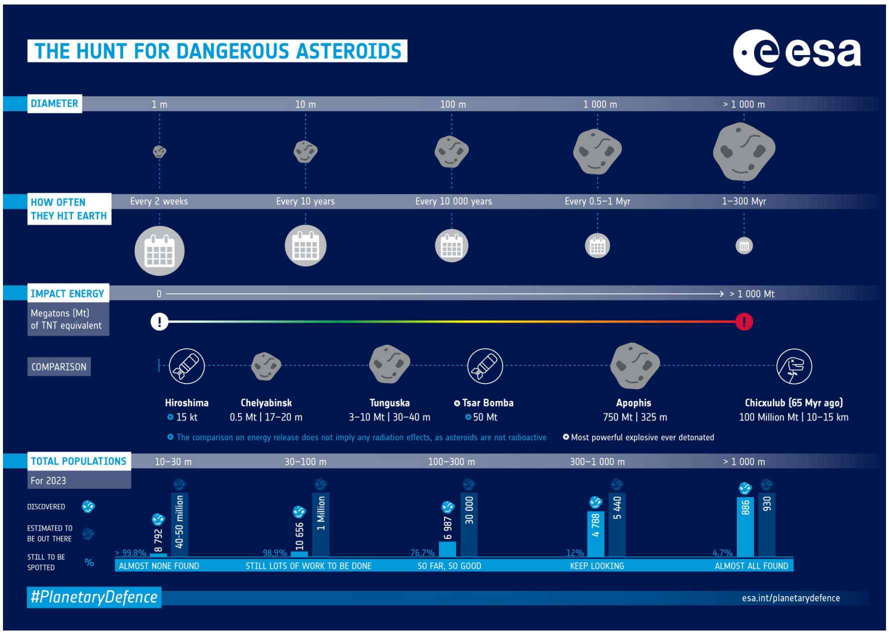


Figure 1.4: Population, energy, and impact frequency of asteroids as a function of the size. Credits: ESA.

This oversight is essential to ensure the spacecraft's sustained functioning and prevent collision with the target body. Navigation accuracy holds particular importance, given these bodies' intricate gravitational fields and irregular shapes. Successful maneuvering and trajectory control to avoid collisions demand precise propulsion systems and navigation algorithms. Moreover, since the spacecraft is often constrained by limited resources, including fuel, power, computational, and data storage capacity, mission planners must carefully balance them to ensure mission success. Traditionally, this has been handled with a ground-based approach, with a certain degree of reluctance to entrust some of these tasks to onboard systems due to issues such as reliability, feasibility, and sub-optimality.

Nonetheless, this existing paradigm has inherent limitations. As the distance from Earth can lead to significant signal delays, real-time spacecraft control cannot be achieved during critical operations such as landing or sample collection. The turn-around time also limits the spacecraft's ability to respond promptly to unexpected scenarios, which are more frequent around an active, mostly unknown, environment as the one around minor bodies. Finally, ground-based operations are expensive and can make up for a substantial portion of the overall cost, curtailing the mission duration and extent of the scientific data gathered.

The effort to overcome these limitations has stimulated the search for alternative strategies, with one promising avenue levaring increasing autonomy for spacecraft operations. This shift would imply delegating more responsibilities to the spacecraft itself, reducing the dependence on real-time commands from Earth and mitigating the issues posed by signal delays.

By incorporating intelligent systems onboard, spacecraft could navigate around complex environments, make split-second decisions, and react to anomalies more adeptly. Such a paradigm shift can enhance mission flexibility, accelerate decision-making, and broaden the scope of scientific observations, all while potentially trimming overall costs. To achieve this, perception of the surrounding environment is the starting point for more complex autonomous operations.

When considering the proximity environment of a small body and all the sensors available on the market, cameras often emerge as the preferred choice due to their lightweight, compact nature, low power consumption, and low cost. As a result, using passive cameras, complemented by image processing algorithms, delivers compelling performance with cost-effective hardware. An increasing variety of image processing techniques are being developed to enable such capabilities in the framework of autonomous optical navigation and comprehension of the surrounding environment about small bodies. Among these, artificial intelligence approaches exhibit the greatest potential, significantly enhancing accuracy while demanding only a fraction of the computational resources compared to conventional approaches.

This transition towards autonomous and intelligent systems is not devoid of challenges, including the need to develop highly reliable methods, the demand for substantial and high-fidelity datasets for training, validation, and testing (a challenge given the limited sample of small bodies that have been thoroughly investigated in the history of space exploration), the unclear certification procedures, and the requirements for execution within the constraints of onboard processors in space, which often exhibit limited absolute performance compared to their terrestrial counterparts.

1.3 Research questions

This work has been carried out in compliance with the activity of the Stardust-R network[6], under the funding received from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 813644.

The current amount of space debris in orbit, combined with the expected increase in traffic due to future mega-constellations, will have an unprecedented impact on the space environment, posing a serious question on its stability and resilience to any incident or anomalous event. Although statistically less likely to occur, an asteroid impact would have devastating consequences for our planet. Thus, Stardust-R addressed the growing need for sustainable exploitation of space, the resilience of the space environment, the threats and opportunities coming from asteroids and comets, and the compelling need for properly trained specialists who

[6]https://cordis.europa.eu/project/id/813644, last accessed 23rd of August 2023.

can tackle these issues. The key scientific objectives of Stardust-R are reported below.

Stardust-R objectives

1. To globally characterize the dynamics of objects around the Earth to define disposal solutions
2. To correlate spatially and temporally distant events and families of debris to their parent object
3. To quantify uncertainty in celestial mechanics to accurately predict the probability of impact and collision and quantify the resilience of space systems and environment
4. To develop AI tools and methods for space traffic management
5. To define a criticality index for small asteroids to identify the need for exploration/characterization, the possibility for exploitation, and the method of deflection
6. To develop a new distribution model for small-size asteroids
7. To develop systems and algorithms to explore and land on minor bodies with autonomous nano-spacecraft.

These objectives have been addressed via *15* projects developed by *15* Early Stage Researchers who have been trained in math, physics, computer science, and aerospace engineering to provide effective solutions to make the space environment resilient and space exploitation sustainable, learn more about minor bodies and ultimately protect Earth and our space assets.

As part of the Stardust-R network, the author has been involved as ESR 13 in the activities of WP7, covering all areas related to the exploration and exploitation of minor bodies. The author was tasked with activities involving autonomous Guidance, Navigation, and Control of low-resource systems for the exploration and exploitation of small bodies. Autonomous deep-space and close proximity navigation, as well as autonomous orbit guidance and control, are the fundamental areas of research by ESR 13 in WP7.

Within the Stardust-R framework, the author focused his research on the development of image processing and autonomous navigation algorithms using artificial intelligence methods. A set of detailed research questions and objectives is elaborated to specifically drive the research presented in this dissertation within the context of the Stardust-R framework.

Research questions

1. Which strategies could be adopted to create the high-fidelity data-label pairs required for the supervised learning of data-driven methods?
   a) What is the most cost-effective and flexible strategy to generate data in support of training, validation, and testing of data-driven algorithms?
   b) To what extent can existing open-source solutions be used to support dataset generation?
2. To what extent can onboard applications benefit from enhanced data-driven image processing methods?
   a) What are the most promising image processing tasks that can be substituted or augmented?
   b) What level of performance can be achieved compared to traditional approaches?
   c) What are the current drawbacks and bottlenecks for their adoptions in real missions?

Research objectives

i) Develop a high-fidelity, easy-to-use, flexible tool for generating image-label pairs to sustain the development of data-driven methods.
ii) Develop a set of techniques that enhance the perception of the surrounding environment using visual images.

1.4 Dissertation overview

This chapter introduces the context, motivation, research questions and objectives, notation, performance metrics, and personal publications of the research activity presented in this manuscript. The methodology section in Chapter 2 illustrates the fundamental theory behind the image processing algorithms, neural networks, and other machine learning applications developed in this research. Chapter 3 represents the design of the critical data generator tools used throughout the manuscript. Depending on the application, image processing methods are split into two main chapters: image segmentation is addressed in Chapter 4, while vision-based navigation is addressed in Chapter 5. Lastly, in Chapter 6, a generic overview of the Milani mission is presented together with the design of the semi-autonomous, vision-based guidance, navigation, and control subsystem of the CubeSat. For clarity, each chapter is closed with its final remarks and recommendations, while some general final comments and future works are discussed in Chapter 7. As data is central in any data-driven model, Appendix A groups all the datasets used in a standardized format. Figure 1.5 illustrates a graphical representation of the relationships between the different chapters and key sections of the manuscript.

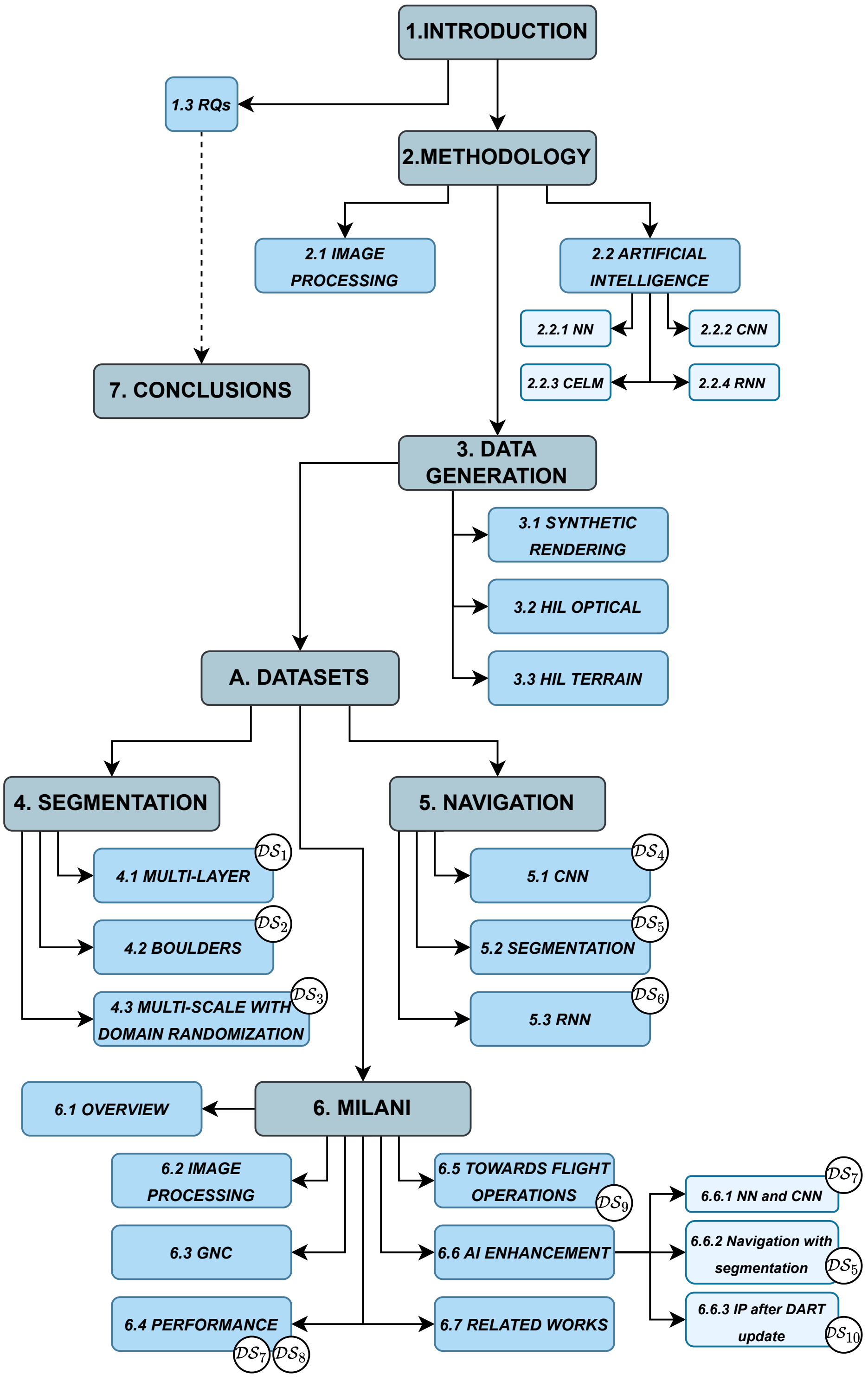


Figure 1.5: Graphical structure of the dissertation illustrating the relationships between different sections of the manuscript. For brevity, only the key sections are represented.

1.5 Publications and contribution to the field

During the years of my Ph.D., I had the chance to present my work at several conferences and to publish in peer-reviewed journals, contributed to several proposals, and actively contributed to several missions and projects related to small bodies exploration, visual-based navigation, and image processing.

Most of the research activity outcome was already presented in various forms (i.e., papers, presentations, technical reports). The list of contributions to the field split among journal articles, conference papers, and datasets is provided below.

Journal articles

[J11] **Pugliatti M**, Buonagura C, and Topputo F. "CORTO: The Celestial Object Rendering TOol at DART lab". In: *Sensors (submitted)* (Sept. 2023).

[J10] Moreno F, Bagatin AC, Tancredi G, Li JY, Rossi A, Ferrari F, Hirabayashi M, Fahnestock E, Maury A, Sandness R, Rivkin AS, Cheng A, Farnham TL, Soldini S, Giordano C, Merisio G, Panicucci P, **Pugliatti M**, Castro-Tirado AJ, Fernández-García E, Pérez-García I, Ivanovski S, Penttila A, Kolokova L, Licandro J, Munoz O, Gray Z, Ortiz JL, and Lin ZY. "Characterization of the Ejecta from the NASA/DART Impact on Dimorphos: Observations and Monte Carlo Model". In: *The Planetary Science Journal* 4.8 (Aug. 2023), p. 138. DOI: 10.3847/PSJ/ace827.

[J9] **Pugliatti M** and Topputo F. "Design and Application of Convolutional Architectures for Vision-Based Navigation Around Small Bodies". In: *Journal of Spacecraft and Rockets (submitted)* (July 2023).

[J8] **Pugliatti M**, Scorsoglio A, Furfaro R, and Topputo F. "Onboard state estimation around didymos with recurrent neural networks and segmentation maps". In: *IEEE Transactions on Aerospace and Electronic Systems* Pre-Print (June 2023), pp. 1–14. ISSN: 0018-9251. DOI: 10.1109/TAES.2023.3288506.

[J7] **Pugliatti M**, Piccolo F, Rizza A, Franzese V, and Topputo F. "The vision-based guidance, navigation, and control system of hera's milani cubesat". In: *Acta Astronautica* 210 (Sept. 2023), pp. 14–28. ISSN: 0094-5765. DOI: 10.1016/j.actaastro.2023.04.047.

[J6] **Pugliatti M** and Maestrini M. "Small-body segmentation based on morphological features with a u-shaped network architecture". In: *Journal of Spacecraft and Rockets* 59.6 (Nov. 2022), pp. 1821–1835. DOI: 10.2514/1.A35447.

[J5] **Pugliatti M**, Franzese V, and Topputo F. "Data-driven image processing for onboard optical navigation around a binary asteroid". In: *Journal of Spacecraft and Rockets* 59.3 (May 2022), pp. 943–959. DOI: 10.2514/1.A35213.

[J4] Peñarroya P, **Pugliatti M**, Ferrari F, Centuori S, Topputo F, Vetrisano M, and Sanjurjo-Rivo M. "Cubesat landing simulations on small bodies using blender". In: *Advances in Space Research* 72.7 (Oct. 2022), pp. 2971–2993. ISSN: 0273-1177. DOI: 10.1016/j.asr.2022.07.044.

[J3] Buonagura C, **Pugliatti M**, and Topputo F. "Image processing robustness assessment of small-body shapes". In: *The Journal of the Astronautical Sciences* 69.6 (Nov. 2022), pp. 1744–1765. DOI: 10.1007/s40295-022-00348-6.

[J2] Ferrari F, Franzese V, **Pugliatti M**, Giordano C, and Topputo F. "Preliminary mission profile of hera's milani cubesat". In: *Advances in Space Research* 67.6 (Mar. 2021), pp. 2010–2029. DOI: 10.1016/j.asr.2020.12.034.

[J1] Ferrari F, Franzese V, **Pugliatti M**, Giordano C, and Topputo F. "Trajectory options for hera's milani cubesat around (65803) didymos". In: *The Journal of the Astronautical Sciences* 68.4 (Sept. 2021), pp. 973–994. DOI: 10.1007/s40295-021-00282-z.

Conference papers

[C23] **Pugliatti M** and Maestrini M. "A multi-scale labeled dataset for boulder segmentation and navigation on small bodies". In: *74th International Astronautical Congress, Baku, Azerbaijan*. Oct. 2023, pp. 1–8.

[C22] **Pugliatti M**, Giordano C, and Topputo F. "The image processing of milani: challenges after dart impact". In: *ESA-GNC conference, Sopot, Poland*. June 2023, pp. 1–15.

[C21] Buonagura C, Borgia S, **Pugliatti M**, Morselli A, Topputo F, Corradino F, Visconti P, Deva L, Fedele A, Leccese G, and Natalucci S. "The cubesat mission future: a preliminary analysis to validate the on-board autonomous orbit determination". In: *ESA-GNC conference, Sopot, Poland*. June 2023, pp. 1–15.

[C20] Giordano C, Ferrari F, Franzese V, **Pugliatti M**, Piccolo F, Rizza A, Kohout T, Dirri F, Longobardo A, C G, Palomba E, Cardi M, Perez-Lissi F, Martino P, and Carnelli I. "The hera milani cubesat mission". In: *5th COSPAR Symposium, 2023*. Apr. 2023.

[C19] **Pugliatti M** and Topputo F. "Boulders identification on small bodies under varying illumination conditions". In: *3rd Space Imaging Workshop, Georgia, Atlanta*. Oct. 2022, pp. 1–12.

[C18] **Pugliatti M** and Topputo F. "Enhanced vision-based algorithms about small bodies: lessons learned from the stardust-r experience". In: *2nd International Stardust conference, STARCON2*. Vol. 1. Nov. 2022, pp. 1–2.

[C17] **Pugliatti M**, Ferrari F, Piccolo F, Rizza A, Bottiglieri C, Franzese V, Giordano C, and Topputo F. "Enhanced vision-based algorithms about small bodies: lessons learned from the stardust-r experience". In: *2nd International Stardust conference, STARCON2*. Vol. 1. Nov. 2022, pp. 1–2.

[C16] Rizza A, Piccolo F, **Pugliatti M**, Panicucci P, and Topputo F. "Hardware-in-the-loop simulation framework for cubesats proximity operations: application to the milani mission". In: *73rd International Astronautical Congress, Paris, France*. Oct. 2022, pp. 1–15.

[C15] **Pugliatti M**, Piccolo F, and Topputo F. "Object recognition algorithms for the didymos binary system". In: *2nd International Conference on Applied Intelligence and Informatics*. Vol. 2. AII, Sept. 2022, pp. 1–20.

[C14] Buonagura C, **Pugliatti M**, Franzese V, Topputo F, Zeqaj A, Zannoni M, Varile M, Bloise I, Fontana F, Rossi F, Feruglio L, and Cardone M. "Deep learning for navigation of small satellites about asteroids: an introduction to the deepnav project". In: *2nd International Conference on Applied Intelligence and Informatics*. Vol. 2. AII, Sept. 2022, pp. 1–20.

[C13] Bottiglieri C, Piccolo F, Rizza A, Giordano C, **Pugliatti M**, Franzese V, Ferrari F, and Topputo F. "Trajectory design and orbit determination of hera's milani cubesat". In: *Advances in the Astronautical Sciences*. Vol. 177. Univelt, Dec. 2021, pp. 81–82. DOI: https://doi.org/10.2514/6.2022-2381.

[C12] **Pugliatti M**, Maestrini M, Di Lizia P, Topputo F, et al. "On-board small-body semantic segmentation based on morphological features with u-net". In: *Advances in the Astronautical Sciences*. Vol. 176. Univelt, Dec. 2021, pp. 603–622.

[C11] **Pugliatti M** and Topputo F. "Navigation about irregular bodies through segmentation maps". In: *Advances in the Astronautical Sciences*. Vol. 176. Univelt, Dec. 2021, pp. 1169–1187.

[C10] Bottiglieri C, Piccolo F, Rizza A, **Pugliatti M**, Franzese V, Giordano C, Ferrari F, and Topptuo F. "Mission analysis and navigation assessment for hera's milani cubesat". In: *4S Symposium*. May 2022, pp. 1–20.

[C9] Buonagura C, **Pugliatti M**, and Topputo F. "Procedural minor body generator tool for data-driven optical navigation methods". In: *6th CEAS Specialist Conference on Guidance, Navigation and Control-EuroGNC*. May 2022.

[C8] **Pugliatti M**, Franzese V, Rizza A, Piccolo F, Bottiglieri C, Giordano C, Ferrari F, and Topputo F. "Design of the on-board image processing of the milani mission". In: *44th AAS Guidance, Navigation and Control Conference*. Feb. 2022, pp. 1–21.

[C7] Piccolo F, **Pugliatti M**, Panicucci P, and Topputo F. "Toward verification and validation of the milani image processing pipeline in the hardware-in-the-loop testbench tinyv3rse". In: *44th AAS Guidance, Navigation and Control Conference*. Feb. 2022, pp. 1–21.

[C6] Panicucci P, **Pugliatti M**, Franzese V, and Topputo F. "Improvements and applications of the dart vision-based navigation test bench tinyv3rse". In: *44th AAS Guidance, Navigation and Control Conference*. Feb. 2022, pp. 1–19.

[C5] **Pugliatti M**, Franzese V, Panicucci P, and Topputo F. "Tinyv3rse: the dart vision-based navigation test-bench". In: *AIAA Scitech 2022 Forum*. Jan. 2022, p. 1193. DOI: https://doi.org/10.2514/6.2022-1193.

[C4] **Pugliatti M**, Rizza A, Piccolo F, Franzese V, Bottiglieri C, Giordano C, Ferrari F, and Topputo F. "The milani mission: overview and architecture of the optical-based gnc system". In: *AIAA Scitech 2022 Forum*. Jan. 2022, p. 2381. DOI: https://doi.org/10.2514/6.2022-2381.

[C3] **Pugliatti M** and Topputo F. "Small-body shape recognition with convolutional neural network and comparison with explicit features based method". In: *Advances in the astronautical sciences*. Vol. 175. Univelt, Aug. 2021, pp. 2539–2258.

[C2] Topputo F, Ferrari F, Franzese V, **Pugliatti M**, Giordano C, Rizza A, Calvi D, Ammirante G, Stesina F, Esposito A, Corpino S, Visconti P, Diaz de Cerio Goenaga R, Corradino F, Santoni A, Cardi M, Kohout T, Perez-Lissi F, Martino P, and Carnelli I. "The hera milani cubesat mission". In: *7th IAA Planetary Defense Conference*. Apr. 2021, p. 163.

[C1] Peñarroya P, **Pugliatti M**, Centuori S, and Topputo F. "Using blender as contact dynamics engine for cubesat landing simulations within impact crater on dimorphos". In: *7th IAA Planetary Defense Conference*. Vol. 2. Apr. 2021.

Datasets

[D3] **Pugliatti M** and Maestrini M. *A multi-scale labeled dataset for boulder segmentation and navigation on small bodies*. Zenodo. Sept. 2023. URL: https://zenodo.org/record/8406581.

[D2] **Pugliatti M**, Giordano C, and Topputo F. *The image processing of milani: challenges after dart impact*. Zenodo. June 2023. URL: https://zenodo.org/record/7962714.

[D1] **Pugliatti M** and Topputo F. *Doors: dataset for boulders segmentation*. Zenodo. Sept. 2022. URL: https://zenodo.org/record/7107409.

1.6 Notation and conventions

The notation and conventions used throughout the manuscript are herewith introduced. Scalars are indicated with lower-case letters (e. g., a). Vectors are indicated with bold lower-case letters (e. g., $\mathbf{p} = [a\ b\ c]^\top$), while their magnitudes with the same letter but in regular font (e. g., $p = \|\mathbf{p}\|$). Estimated quantities are denoted with the diacritic ^{e}, while true ones are with ^{t}.

A standardized notation is used across the manuscript to describe the layers and structure of the neural networks. For simplicity, the notation makes use of acronyms and naming conventions used in TensorFlow (TF) 2.10. The most popular layers used are summarized in Table 1.3. Specific notations different than the one illustrated in this section will be introduced on an as-needed basis.

Table 1.3: Notation used to identify the layers of the architectures in this manuscript.

ID	Short name
I	Input
O	Output
D	Dense
C	Conv2D
CT	TransposeConv2D
UP	Upconvolution (Sequential)
E	Encoder (Sequential)
L	Long-Short Term Memory
F	Functional (Sequential)
CC	Concatenate
FC	Flatten
A	Generic activation
LR	LeakyReLU
R	ReLU
P	Pooling2D
DO	Dropout

1.7 Reference frames

The most used reference frames are briefly defined hereafter. $\mathcal{W}$ is a sun-oriented reference frame defined as:

- **Origin**: Centered on the Center of Mass (CoM) of the target body.
- **X-axis**: Oriented towards the projection of the Sun in the target body's equatorial plane.
- **Y-axis**: Follows from the definition of the X and Z axes.
- **Z-axis**: Coindient with the north pole of the target body.

$\mathcal{AS}$ is a body-fixed reference frame defined as:

- **Origin**: Centered on the CoM of the target body.
- **X-axis**: Defines the equatorial plane with the Y axis.
- **Y-axis**: Follows from the definition of the X and Z axes.
- **Z-axis**: Coindient with the north pole of the target body.

$\mathcal{UV}$ is an image-based reference frame defined as:

- **Origin**: Centered on top-left pixel of the image
- **U-axis**: Horizontal axis that spans the columns of the image.
- **V-axis**: Vertical axis that spans the rows of the image.

$\mathcal{CAM}$ is a body-fixed reference frame defined as:

- **Origin**: Centered on the principal point of the sensor.
- **X-axis**: Oppositely aligned with the V axis of the $\mathcal{UV}$ reference frame.
- **Y-axis**: Aligned with the U axis of the $\mathcal{UV}$ reference frame.
- **Z-axis**: Represents the camera boresight, outgoing from the camera.

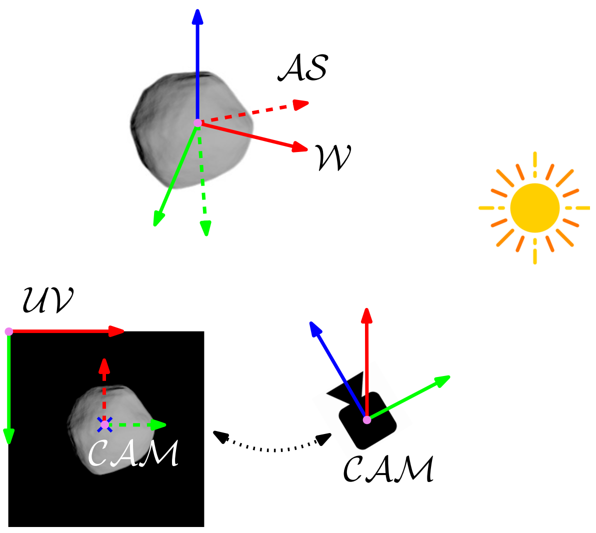


Figure 1.6: Simple schematic of the reference frames used in this manuscript. The X, Y, and Z axes are represented respectively in red, green, and blue for each reference frame.

1.8 Performance metrics

For simplicity, the most common performance metrics considered in this manuscript are reported here. Specific metrics are introduced contextually on an as-needed basis.

1.8.1 Regression

To assess the performance of Image Processing (IP) algorithms with respect to regression tasks, the following metrics are considered:

$$\varepsilon^{u}_{CoF} = CoM_u - CoF_u \quad ; \quad \varepsilon^{v}_{CoF} = CoM_v - CoF_v \tag{1.1}$$

$$\varepsilon^{n}_{CoF} = \sqrt{\left(\varepsilon^{u}_{CoF}\right)^2 + \left(\varepsilon^{v}_{CoF}\right)^2} \tag{1.2}$$

where CoM represents the true CoM while CoF represents the estimated one, also referred to as Center of Figure (CoF), ε^{u}_{CoF} and ε^{v}_{CoF} are the centroid pixel error by coordinates in the image plane, and ε^{n}_{CoF} is the distance in pixel in the image plane between the estimated and true CoM.

$$\varepsilon_{\alpha} = \varepsilon^{n}_{CoF} \cdot \frac{\zeta}{AS} \cdot 100 \tag{1.3}$$

$$\varepsilon_{m} = 2\rho^{t} \cdot \tan\left(\frac{\zeta \cdot \varepsilon^{n}_{CoF}}{2}\right) \tag{1.4}$$

where AS is the apparent size of the target body, ρ^t is the true range from the target body, and ζ is the instantaneous Field Of View (FOV). ε_{α} represents the ε^{n}_{CoF} error expressed as a relative percentage error in angular size with respect to the size of the target body and ε_m represents the projection of the ε^{n}_{CoF} error in the image plane expressed in m.

$$\varepsilon_{\psi} = \psi^{e} - \psi^{t} \tag{1.5}$$

$$\varepsilon_{\rho} = \rho^{e} - \rho^{t} \tag{1.6}$$

where ρ^t and ψ^t, and ρ^e and ψ^e indicate respectively the true and estimated range and phase angle from the target body, being ε_{ψ} and ε_{ρ} the absolute errors of such quantities.

1.8.2 Object recognition

To assess the performance of the IP algorithms for object recognition and classification tasks, the following metrics are considered:

$$A = \frac{TP + TN}{TP + FP + TN + FN} \tag{1.7}$$

$$P = \frac{TP}{TP + FP} \tag{1.8}$$

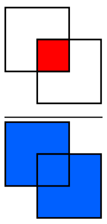
$$R = \frac{TP}{TP+FN} \tag{1.9}$$

where A, P, R stand respectively for accuracy, precision, and recall and TP, TN, FP, FN stands respectively for *True Positive*, *True Negative*, *False Positive*, and *False Negative*.

1.8.3 Semantic segmentation

To assess the performance of the IP algorithms for semantic segmentation tasks, the following metrics are considered:

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}} = \tag{1.10}$$



We then define the mean Intersection over Union (mIoU) as the average of the Intersection over Union (IoU) computed for each class. The former represents a global segmentation performance metric for each image or dataset, the latter a metric for each specific layer.

$$mIoU = \frac{\sum_{i=1}^{n_{masks}} IoU_i}{n_{masks}} \tag{1.11}$$

1.8.4 Neural networks performance

To assess the performance of a neural network, the following metrics and loss functions are considered:

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |\hat{y}_i - y_i| \tag{1.12}$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N} (\hat{y}_i - y_i)^2 \tag{1.13}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (\hat{y}_i - y_i)^2} \tag{1.14}$$

where $\hat{y}_i$ represents the predicted values, y_i the true or target values, and N is the number of samples, and the metrics are referred to as Mean Absolute Error (MAE), Mean Squared Error (MSE), and RMSE. For classification tasks, it is also useful to define:

$$SSCE = -\sum_{i=1}^{N} y_i \log \hat{y}_i \tag{1.15}$$

$$WSSCE = -\frac{1}{M}\sum_{j=1}^{M}\sum_{i=1}^{K} w_j y_{i,j} \log \hat{y}_{i,j} \tag{1.16}$$

where w_j represents the jth component of the weight vector associated with a specific class, and Sparse Categorical Cross Entropy (SCCE), and Weighted Sparse Categorical Cross Entropy (WSCCE).

1.8.5 Navigation and pointing

The performance of the navigation and pointing strategies are evaluated using the following metrics:

$$\varepsilon_{\mathbf{p}} = \mathbf{p}^e - \mathbf{p}^t \tag{1.17}$$

$$\varepsilon_{\mathbf{v}} = \mathbf{v}^e - \mathbf{v}^t \tag{1.18}$$

$$\varepsilon_{\theta} = \text{acos}(\mathbf{u}_{SC} \cdot \mathbf{u}_B) \tag{1.19}$$

where $\mathbf{p}^e$, $\mathbf{v}^e$ and $\mathbf{p}^t$, $\mathbf{v}^t$ are respectively the estimated and true position and velocity components of the spacecraft state vector, $\mathbf{u}_{SC}$ and $\mathbf{u}_B$ are respectively the Line of Sight (LoS) of the spacecraft's payload and the LoS to the CoM of the target body. $\varepsilon_{\mathbf{p}}$ and $\varepsilon_{\mathbf{v}}$ represent the position and velocity estimation errors, while ε_{θ} represents the pointing error. Finally, relative position and velocity errors are defined as:

$$\varepsilon_{\mathbf{p}}^r = \frac{\|\mathbf{p}^e - \mathbf{p}^t\|}{\|\mathbf{p}^t\|} \cdot 100 \tag{1.20}$$

$$\varepsilon_{\mathbf{v}}^r = \frac{\|\mathbf{v}^e - \mathbf{v}^t\|}{\|\mathbf{v}^t\|} \cdot 100 \tag{1.21}$$

Acknowledgments The work presented in this dissertation has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 813644.

Part of the research on the Milani CubeSat mission presented in Chapter 6 has been conducted under European Space Agency (ESA) Contract No. 4000131925/20/NL/GLC for the phases A/B/C/D/E. The author also acknowledges the members of the Milani consortium for their support and the ESA experts for their review and feedback.

Part of the research on the rendering tools presented in Chapter 3 has been performed as part of the DeepNav project under ASI contract No.2021-16-E.0. The author also acknowledges the members of the DeepNav consortium for their support and the Agenzia Spaziale Italiana (ASI) experts for their review and feedback.

Finally, the author would like to thank the valuable and constructive comments of the reviewers, which have been crucial in improving the quality of the final manuscript.

Methodology

"Artificial Intelligence: the silent evolution of ones and zeros, where algorithms whisper secrets of innovation, and data paints the canvas of progress, revealing the future in lines of code."

ChatGPT-3.5, answering to the prompt: *"Write a great quote about AI"*

Computer vision refers to the action of acquiring, processing, analyzing, understanding, and extracting high-dimensional data from the real world sensed via digital images to produce information [21]. Within this context, understanding means transforming a visual image into a description of the world that makes sense and can elicit an appropriate action.

Amazingly, humans and animals do this effortlessly, while computer vision algorithms are error-prone. Indeed, it is straightforward to develop a mathematical description of the problem that either does not match realistic world conditions or proves unreliable. Vision is so difficult partly because it represents an inverse problem: fully specifying the solution by recovering some unknowns, given insufficient information of the world sensed through images [22].

A complete and exhaustive history of computer vision and its dominant trends in each decade can be found in [22], from which the following extracts are considered. When computer vision started in the early *1970*s, it was framed as part of an ambitious agenda to mimic human intelligence. It was believed that solving "the visual input" problem would be an easy step along the path to solving more difficult ones. According to one well-known story, in 1966, Marvin Minsky at MIT asked his undergraduate student Gerald Jay Sussman to "spend the summer linking a camera to a computer and getting the computer to describe what it saw" [22, 23]. After decades, a complete solution to this puzzle remains elusive as a testament to the complexity of the task. A robust trend emerged in the *2000*s that would forever transform the field, embracing data-driven methods and learning approaches as core components in computer vision algorithms. This trend became a "tidal wave" in the *2010*s due to various favorable factors. This change was primarily fueled by the availability of large-scale, high-quality annotated datasets, and the dramatic increase in computational power from general-purpose algorithms on Graphic Processing Unit (GPU). Secondarily, the rapid dissemination of ideas through timely publications on arXiv[7], the development of languages and libraries for deep learning methods, and the open-source nature of many neural network models contributed to explosive growth in this area [22]. These elements are often cited as the three pillars that sustained the Artificial Intelligence (AI) revolution for modern computer vision applications. Albeit the progress made, the dream of achieving a computer's ability to describe an image with the same level of intricate detail and causal understanding as a two-year-old is still challenging to achieve [22].

When considering space applications, a new set of challenges arises on top of the traditional ones. The low-resolution sensors typically used in navigation cameras, coupled with the limited computational, memory, and data capabilities of space-qualified hardware, impose significant limitations on approaches that otherwise flourish in the classical computer vision domain. For these reasons and to a certain degree, image processing algorithms for space applications are some decades behind in terms of development compared to traditional domains.

For example, referring to the three pillars that sustained the AI revolution in modern computer vision, high-fidelity, large-scale, open-access annotated datasets are still difficult, if not impossible, to find. At the same time, the capabilities of radiation-hardened hardware severely undermine the implementation of conventional deep architectures, which have been investigated for space applications only in recent years with Machine Learning (ML) approaches [1, 24, 25].

Compared to traditional methods, in which features and algorithms are hand-crafted, enhanced image processing pipelines implement elements of AI, boosting performance. The explainability and control over a traditional pipeline's steps come at the cost of sub-optimal performance. Enhanced image processing pipelines, on the other hand, are tuned through data and thus can challenge traditional assumptions, opening up the design search space of the algorithm. These enhanced methods can be divided into two groups, as illustrated in Figure 2.1: hybrid and end-to-end ones. In a hybrid pipeline, only the algorithm is developed including elements of AI, while the features used to represent images are still hand-crafted. In an end-to-end pipeline, both features and models are learned through the data, a coupling that often provides the best performance. These enhancements come at the cost of having annotated data representative of real-world conditions for training.

The rest of the chapter is structured in two main parts. First, in Section 2.1, a generic overview of the traditional image processing methods specifically for space applications around minor bodies is provided. Then, in Section 2.2, the fundamental theoretical blocks about ML used in this manuscript are detailed.

[7]https://arxiv.org/, last accessed 31st of August 2023

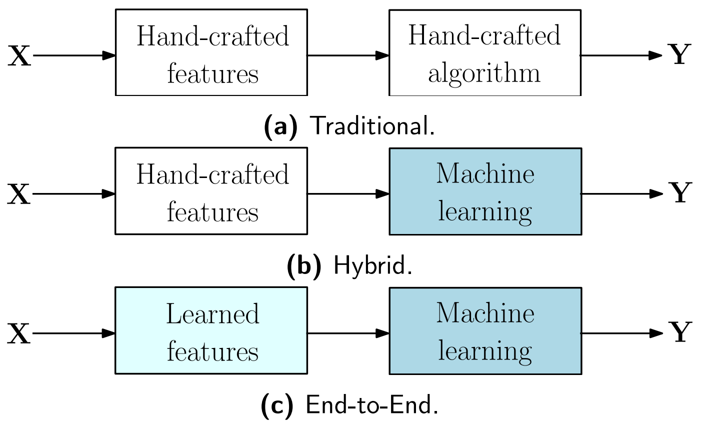


Figure 2.1: (a) Traditional, (b) hybrid, and (c) end-to-end pipelines. **X** and **Y** represent respectively the input and the output. Adapted from [22].

2.1 Image Processing around small bodies

Since the dawn of space exploration, onboard cameras have been extensively used for a variety of tasks: as means of scientific investigations to enabling sensors for precise targeting, navigation, hazard determination, and public outreach. However, it is only when the data produced by these sensors is made available onboard, after being processed by an IP algorithm, that a spacecraft can exploit them to increase its level of autonomy.

Several IP algorithms have been deployed in various exploration missions towards small bodies. Considering the design of these algorithms and the body's appearance in the image as the main property, it is simpler to divide the region of space around small bodies into three relative regimes: far, medium, and close. These regimes reflect in an incremental order the vicinity to the target body, the difficulty of the environment, and the complexity of the IP method. For this reason, they have also been historically approached in this order, as prudent approaches were preferred in previous missions, with an incremental increase of the risk over the years as technology improved, allowing spacecraft to get closer safely.

The far regime extends roughly from the first detection of the body to the moment it occupies a relatively small area imaged by the sensor. In this regime, no global structures or local features would be visible. In the medium regime, the body would start to occupy a considerable portion of the sensor, exhibiting clear global features (shape outline, large craters, or large geological formations altering the surface's appearance) but still lacking the presence of local features such as smaller craters and boulders. As soon as the body's appearance saturates the camera's FOV, global features cease to be imaged in their entirety while detailed representations of the surface are detected. Entering the close regime, local features such as small craters, boulders, and other minor geological features are observed at high resolution. This regime eventually merges into surface operations for any robotic system designed to be stationary or move across the body's surface. Designing a vision-based autonomous system for a mission often means a traverse through these regimes, combining different methods at different stages of a mission. Examples of the same target body seen at different regimes are illustrated in Figure 2.2. For a more comprehensive overview of the requirements and needs of different tasks in these regimes, the reader is redirected to [26].

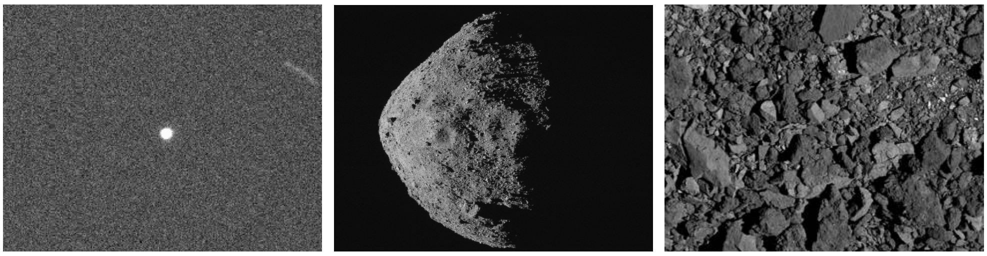

(a) Far. **(b)** Medium. **(c)** Close.

Figure 2.2: Example of Bennu seen from OSIRIS-REx in the far (a), medium (b), and close (c) regimes. Credits: NASA/OSIRIS-REx team.

What follows is an overview of the state-of-the-art approaches across the regimes, both with traditional and enhanced IP methods. The overview mainly focuses on image processing for segmentation and visual-based navigation, as these are the main tasks investigated in this manuscript with enhanced techniques.

2.1.1 Far regime

This regime is the only one accessible from Earth-based and space-based telescopes. Lightcurve analysis and simple centroid methods are used to track the body amongst stars and planets [26]. As the body starts appearing at sub-pixel level, no meaningful information other than centroid and lightcurve is extracted. The main task that can be performed in this regime is to keep track of the target body, distinguishing from other celestial bodies and noise in the sensor to maintain a stable approach in preparation for the medium regime.

2.1.2 Medium regime

Various techniques can be used in the medium regime as the body develops distinguishable global features in the images.

From a navigation point of view, centroiding algorithms are considered as a robust, easy-to-use baseline both for rendezvous [27], flyby [28], and deflection [29] missions in this regime. The simplicity of these algorithms generally comes at the cost of low performance, especially when considering high-phase angles and highly irregular shapes [30, 31]. Over the years, modified centroid algorithms have also been developed to overcome these limitations by adopting analytic scattering functions [31–33].

Another approach to cope with high phase angles is represented by centroid and apparent diameter methods that use the body's limb. The state-of-the-art algorithm of this family is the one introduced in [34], which develops a non-iterative ellipse fitting to retrieve the position with respect to a body modeled as a tri-axially ellipsoid. Other important works exist based on limb and ellipse fitting, such as [30, 35–39] or Lambertian sphere correlation algorithms [40].

By design, the applicability of these methods is limited to spherical-like bodies only, which is a critical disadvantage considering that most of the minor bodies are highly irregular. Attempts have been made to assess ellipse fitting performance with irregular bodies [30, 35]. On the other hand, correlation methods that use information from the global outline exhibited good performance even with mildly irregular shapes [40] correlating to spheres and using rough shape models [41, 42].

Deep learning-based methods for relative navigation are becoming increasingly relevant for this regime [1, 24, 25], in particular related to IP, vision-based navigation, and control methods. Navigation applications have been recently proposed for AI enhanced centroiding or position estimation networks considering irregular bodies [32, 43–45]. Within this area, a variety of works is emerging concerning relative pose estimation from images [46] with deep neural networks, enhanced centroid and apparent diameter methods [47], image-based guidance for landing applications on the Moon [48, 49], and position estimation around small bodies from images [50].

Finally, from a purely IP perspective, a variety of algorithms have been developed to detect, match, and track image's features generated by morphological properties for shape reconstruction purposes around a small-body [26, 51, 52] from the end of the far to the end of the medium regimes.

Fewer methods have focused so far on image segmentation to discern different morphological features over the surface of a small body. Image segmentation has been used to distinguish between plumes and jets from comets and moons in [53], combining simple pixel-intensity-based methods and geometric considerations. In [54], a series of advanced image processing techniques for enhanced flyby science around small bodies are introduced to change the current flyby paradigm and include a more comprehensive understanding of the environment onboard a spacecraft, ultimately strengthening the scientific outcome of flyby missions. The authors in [54] present a methodology for autonomous feature detection supported by simple filtering and statistical-based classification alongside image segmentation to distinguish between features, surface, and background pixels.

Deep learning, in the form of simple and easy-to-train UNet architectures, revolutionized image segmentation. In [49, 55–57] these are used to synthesize hazard maps for selecting safe lunar landing sites. In [58] a thorough comparison between different architectures is presented for hazard detection. The accuracy of deep-learning-based methods is also highlighted in [59], which uses a set of five different Convolutional Neural Network (CNN) architectures to detect geological structures at varying scales for the Mars Reconnaissance Orbiter mission, demonstrating that their methodology outperforms other states of the art methods used previously for the same task and that they could be implemented for onboard applications.

A more comprehensive and up-to-date survey of deep-learning methods for space applications can be seen in [24].

2.1.3 Close regime

Finally, considering navigation, features-based methods are usually preferred in the close-range regime due to the abundance of resolved local features and higher performance compared to the other techniques presented in the previous sections.

For example, the optical navigation approach of the Rosetta mission, presented in [60], used small-scale, high-resolution digital models called maplets centered on specific landmarks of interest scattered across the surface of 67P as reference points for precise local correlation methods. This approach relied on the simulation of each maplet at varying illumination, elevation, and albedo modeling. By correlating these digital representations with their real counterpart extracted from images, the spacecraft's position is estimated as the one maximizing the correlation score. It is noted that this approach has not been executed onboard but heavily relied on a man-in-the-loop process performed from the ground [60].

A similar approach has instead been developed specifically for an onboard implementation in the OSIRIS-REx mission [61, 62], exploiting natural features rendered and correlated thanks to detailed maps generated carefully in previous mission phases. This technique, called Natural Feature Tracking (NFT), represents the most advanced form of feature-tracking that has flown and applied to a small body up to date. On the other hand, relative feature tracking algorithms based on the Kanade-Lucas-Tomasi method described in [63] in combination with navigation filters will be adopted in Hera for the most critical phases in the vicinity of Didymos [40], complementing Lambertian sphere correlation techniques used from further distances.

Another interesting approach is instead adopted by the Hayabusa-II mission, using artificial landmarks dropped on the surface of the body as reference beacons [64]. Being the landmarks retro-reflective, they are easily detected by the cameras after being artificially stimulated by stroboscopic lights timed before image acquisition. Compared to natural landmarks, feature recognition and matching are significantly easier.

Finally, many other works in the literature exist regarding absolute and relative feature-based methods [24, 65–72] using boulders, craters, or other sort of local features.

For what concern IP and segmentation, typically, the works presented in the previous section find applications both in the medium and close regimes. The main challenge is the capability to robustly detect features such as boulders and craters at different scales (given that these landmarks often exhibit fractal appearance) and under strongly varying illumination conditions and occultations.

2.2 Artificial Intelligence

In its broadest definition, AI is the science of developing machines that can act and make decisions "intelligently". As illustrated in Figure 2.3, it encompasses both ML, defined as the capability to train a model to perform tasks without explicitly programming them, and Deep Learning (DL), considered as the most sophisticated form of ML that employs deep neural architectures that try to mimic the functioning of biological brains. It is important to distinguish between these terms, as they are often misused.

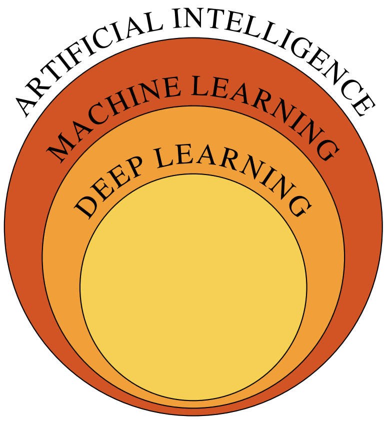


Figure 2.3: AI, ML, and DL are different concepts in relation to each other as three concentric circles.

When considering the concept of learning, there are three distinct paradigms:

- **Supervised**: A model learns from annotated input-output relationships. The annotations, referred to as labels, are actively used by the model to learn an input-output representation. Classification and regression tasks are often addressed via supervised learning approaches. The drawback of this approach is the need for annotated datasets, which, depending on the domain, could be expensive.
- **Unsupervised**: A model is tasked to find patterns, structures, or relationships in unlabeled data. In this form of learning, the model learns from the data without explicit supervision or pre-defined labels. The main goal of this strategy is to discover hidden patterns, clusters, or representations hidden within the data. Clustering, association, and dimensionality reduction are typical unsupervised tasks.
- **Reinforced**: A model, referred to as an agent, learns to take a sequence of actions within the environment to maximize a cumulative cost function via positive or negative rewards. Instead of relying on labeled data, it depends on trial and error of the agent taking actions within the environment, observing the resulting state and reward to improve its decision-making policy. Game playing, robotics, and autonomous systems often use this form of learning.

These three paradigms represent distinct approaches to machine learning, each suited to different types of problems and data. The one used in this manuscript is related to supervised learning.

In supervised learning, various models and architectures can be used for classification, regression, and segmentation tasks investigated throughout the manuscript. The theory behind the main architectures used in this research is illustrated in detail in an incremental order of complexity in the next sections.

2.2.1 Neural Networks

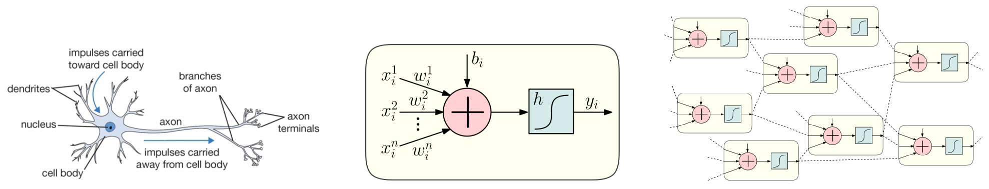
This section illustrates the core concepts used to properly design and train a neural network. The concepts presented here are the bases that apply also to other forms of more complex architectures illustrated toward the end of the chapter.

2.2.1.1 Neurons, weights and biases

A neuron is the fundamental unit at the base of neural networks. Its functioning is pretty simple since, in its simplest form, the $i-th$ neuron produces an output based on a biased and weighted sum of its inputs followed by an activation function re-mapping [22]:

$$y_i = h\left(\mathbf{w}_i^T \mathbf{x}_i + b_i\right) \tag{2.1}$$

where $\mathbf{w}_i^T$ and b_i represent the learnable weights and bias of the neuron, $\mathbf{x}_i$ is the vector collecting all of the inputs, and h is a non-linear activation function. A schematic of this simple operation is also illustrated in Figure 2.4 and is loosely based on the functioning of real neurons in biological systems



(a) Biological neuron. **(b)** Artificial neuron. **(c)** Artificial neurons.

Figure 2.4: Schematic of a single biological neuron (a) from [8], single artificial neuron (b) and irregular computation graphs with multiple neurons (c).

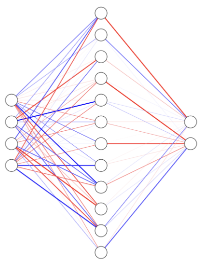
Having defined its most fundamental unit, a Neural Network (NN) can be represented as a computation graph composed of interconnections between neurons. Such structure is inspired by similar interconnection between neurons in biological brains. One such structure is illustrated in Figure 2.5. However, instead of being organized as an irregular computational graph, NNs are usually organized into regular consecutive layers. From a mathematical perspective, this is useful for

[8]https://cs231n.github.io/neural-networks-1/, last accessed 30th of August 2023.

establishing a simple relationship between all the neurons within a layer and its output:

$$\mathbf{y_l} = h_l\left(\mathbf{W}_l \mathbf{x}_l\right) \tag{2.2}$$

where $\mathbf{x}_l$ are the input of the layer, $\mathbf{W}_l$ is a weight matrix representing the weights between each layer's input to each neuron, and h_l is the layer's activation function, assuming all the neurons in that layers share the same activation function. Layers in which a full weight matrix $\mathbf{W}_l$ is used are called "dense" or "fully connected" layers since all of the inputs to one layer are connected to all of its outputs. A representation of different NN regular architectures is illustrated in Figure 2.5.


(a) Single layer NN.

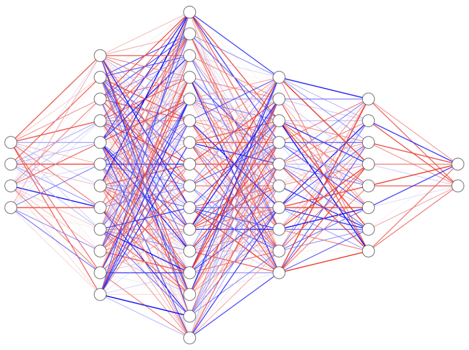

(b) Multi-layers NN.

Figure 2.5: (a) Single layer and (b) multi-layers NN architectures. The networks are initialized with random weights and biases. The opacity of the connections represents the weight norm, while different colors represent the sign (negative for red and positive for blue). Generated from [9].

In its simplest form, a NN can even take the structure of a single neuron as illustrated in Figure 2.4(b), as has been the case for the perceptron illustrated in [73] for binary classification. More complex tasks also demand complex architectures. An architecture is defined as a single-layer feedforward network or multi-layer perceptron when, as illustrated in Figure 2.5(a), it is composed only of one fully connected layer between input and output. When multiple layers are used, referred to as hidden layers, the network is defined as a shallow or deep network, depending on the number of hidden layers. An example of shallow, multi-layer architecture is illustrated in Figure 2.5(b).

The entire architecture's weights and biases define the set of parameters grouped by θ, also referred to as internal or local. The design of the architecture and parameters used to train a model defines the set of parameters grouped by $\mathbf{\Theta}$, also referred to as external, global, or hyperparameters of the model. θ are optimized

[9] https://alexlenail.me/NN-SVG/index.html, last accessed 1st of September 2023

during training, $\mathbf{\Theta}$ are the results of arbitrary choices about the architecture or output of an extensive search grid. A trained architecture can be used in inference as a function π (parameterized by $\mathbf{\Theta}$ and θ) that acts on the input $\mathbf{X}$ to generate the output vector $\mathbf{Y}$:

$$\mathbf{Y} = \pi_{\Theta}(\mathbf{X}|\theta) \tag{2.3}$$

The optimal sets of θ and $\mathbf{\Theta}$ may be denoted by a * superscript and are found via the training and validation

2.2.1.2 Activation functions

Various activation functions are defined in the literature and used in NN. Amongst them, the most used ones throughout the manuscript are:

$$Sigmoid = \frac{1}{1+e^{-x}} \tag{2.4}$$

$$Tanh = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2.5}$$

$$ReLU = \begin{cases} x & \text{if} \quad x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

$$LReLU = \begin{cases} x & \text{if} \quad x > 0 \\ \alpha x & \text{otherwise} \end{cases} \tag{2.7}$$

$$Softmax = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{2.8}$$

The output of the sigmoid and tanh are bounded respectively between 0 and 1, and -1 and 1. The ReLU is bounded between 0 and +inf, while the LReLU is unbounded. Finally, the softmax is an activation function that values into probabilities. The softmax output is a vector representing each possible outcome's class likelihoods. For its property, the softmax is often used as an activation function in the last layer of classification networks used for classification.

The choice of the activation function impacts training time and the robustness of the network. Each function has its pros and cons, which this section does not discuss further. ReLU is often used as a preferred function in convolutional architectures due to its simplicity and low computational cost during training. However, findings from [74] and from experience hinted that LReLU performs better than ReLU when considering noisy images while avoiding irreversible neurons dying during training, which can happen with poor initialization. Thus, whenever the choice of the activation function is not part of the hyperparameter search, in this manuscript, either the ReLU or LReLU will be primarily used as the activation functions.

2.2.1.3 Loss and metric functions

The loss function is fundamental to compute the mismatch between predicted and actual labels in a supervised learning setting. As for the activation function, various loss functions exist depending on the specific task required. Some have already been introduced in Section 1.8.4. As discussed in the following sections, the loss function plays a pivotal role in the training of the network and, thus, must satisfy specific mathematical properties regarding its differentiability. A metric function is instead used to evaluate the performance of a model without being directly involved in the training.

2.2.1.4 Backpropagation

Using the building blocks presented in the previous sections, it is possible to assemble a rudimental neural network. Such a network, however, would be practically useless without an algorithm to adjust its weights and biases, orienting them in a way that minimizes the loss function.

The most successful, known, and used form of training relies on the backpropagation algorithm, first introduced in [75]. Its simple steps are briefly described below.

- **Step 1** ***(Forward pass)***: Training data is propagated through the network, which is instantiated by a set of weights $\mathbf{w}$ and biases $\mathbf{b}$. This pass produces outputs for each network layer, from input to output.
- **Step 2** ***(Weights and Bias adjustment)***: weights and biases are adjusted to reduce the network error.
  - **Step 2.1** ***(Error computation)***: The difference between the predicted output layer and its expected value is quantified with the use of the loss function L.
  - **Step 2.2** ***(Backward pass)***: For each set of $\mathbf{w},\mathbf{b}$, the derivative $\mathbf{g}_t$ of the error is found with respect to them. The computation of $\mathbf{g}_t$ starts from the back of the network (the output layer) and then moves backward toward the start of the network (the input layer). This is because $\mathbf{g}_t$ is simple to compute for the final output layers, which is directly linked to L. $\mathbf{g}_t$ can then be computed at each layer using the next layer's derivative computation, passing from the back to the start of the network. The algorithm takes its name from this passage, which can be seen as the backpropagation of the error, which is possible by applying the chain rule.
  - **Step 2.3** ***(Weights and Biases update)***: The values of $\mathbf{w},\mathbf{b}$ are adjusted by a negative proportional value of the derivative value $\mathbf{g}_t$ (for example, for $\mathbf{w}$ this corresponds to $\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha_t \mathbf{g}_t$, where α is a step size parameter called learning rate.
- **Step 3** ***(Repeat)***: The process is repeated from step 1 with a new set of $\mathbf{w}_{t+1},\mathbf{b}_{t+1}$.

Different strategies can be used as stop criteria of this iterative algorithm: a fixed number of iterations, the value of the loss function on the validation set, or convergence of the weights and biases.

The computation of the derivatives g_t requires the unit activations computed in the forward pass. However, a typical implementation of a NN could be using millions of units whose output values must be stored and used during training.

2.2.1.5 Training

The iterative process illustrated in the previous section is at the heart of the usual training procedure of NN. Depending on the size $\mathcal{B}$ of training data flowing at each iteration, typically, three main variations of gradient descent algorithms are distinguished: Stochastic Gradient Descent (SGD), Mini-Batch Gradient Descent (MBGD), and Gradient Descent (GD).

In the SGD, a single training sample ($\mathcal{B} = \mathit{1}$)is used to evaluate the loss function and compute the derivatives g_t. However, the directions obtained using the SGD are characterized by noisy updates. To avoid this, losses and gradients are usually summed over a small subset of the whole training data grouped into a batch $\mathcal{B} = M$ in what is referred to as MBGD:

$$L_{\mathcal{B}}(\mathbf{w}) = \sum_{M \in \mathcal{B}} L_m(\mathbf{w}) \tag{2.9}$$

$$g = \nabla_{\mathbf{w}} L_{\mathcal{B}} \tag{2.10}$$

where $\mathcal{B}$ is the batch or mini-batch, and M is its size, comprised between $\mathit{1}$ and N, the latter being the entire dataset size. Finally, if $\mathcal{B} = N$, we refer to the method as GD.

The MBGD is often the preferred choice since it combines frequent weights and biases updates with reasonable noise levels during gradient descent. During training, the batch size $\mathcal{B}$ and learning rate α constitute important parameters [76]. Learning rate schedulers can also be implemented at variable speeds to allow training to start with larger values that decrease over time, allowing the optimization to settle quickly into a minimum [22].

Moreover, using batches makes it possible to load and process smaller inputs, decreasing the computational load and memory while increasing convergence speed, both of which are issues of the SGD and GD, respectively. The batch size is thus often a critical design choice when training a CNN.

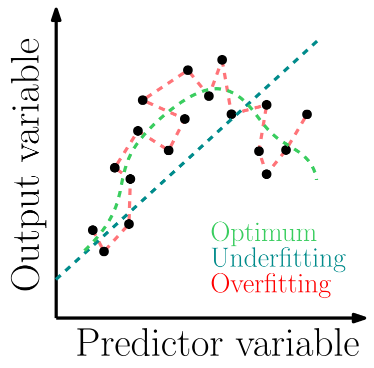
Traditional gradient descent algorithms are prone to stall when reaching a flat region in the search space. For these reasons, momentum-based algorithms have been implemented as extensions of gradient-descent ones. The momentum is implemented by an exponentially decaying running average of the gradient, which is compounded and provides a direction update:

$$\mathbf{v}_{t+\mathit{1}} = \rho \mathbf{v}_t + \mathbf{g}_t \tag{2.11}$$

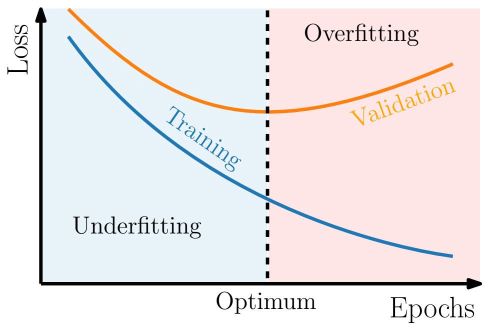
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha_t \mathbf{v}_t \tag{2.12}$$

Many variations of these optimization schemes have been developed in the last decades. The most popular ones are *Nesterov momentum*, *Adaptive gradient AdaGrad*, *RMSProp*, *Adadelta*. However, the most used one [22] is currently *Adam*, which groups elements of all the previous schemes into a unified framework. The peculiarities of these algorithms are not described in detail in this manuscript.

Important undesirable behaviors to prevent during training are underfitting and overfitting, illustrated in Figure 2.6.



(a) Output-Prediction.



(b) Training episode.

Figure 2.6: Underfitting and overfitting from an output-prediction point of view (a) and from a loss curve behavior point of view (b).

Overfitting is an undesirable behavior occurring when a model gives accurate predictions only for training data. It happens because the model has sufficient complexity to memorize the training data, embedding a representation within the weights and biases of the network. An overfit model cannot generalize to unseen data, making it unreliable for inference. Underfit is the opposite phenomenon, in which a too-simple model cannot learn a fruitful representation of the data. Underfit can result from a model needing more time for training, more features to consider as input, or less regularization put in place against overfit. A balanced network reaches an equilibrium between these opposite behaviors.

As illustrated in Figure 2.6(b), underfit and overfit are not visible from the behavior of the training loss, but they are instead clear from looking at the behavior of the validation loss, computed over samples that are not used during training.

Finally, the typical training strategy with gradient-based methods is schematized in Figure 2.7.

It is good practice to divide the dataset into three different splits of variable composition referred to as train, validation, and test sets. The model uses the training set for learning and optimizing θ. Concurrently, during this phase, a validation set can be used to tune the hyperparameters $\mathbf{\Theta}$ and to avoid underfitting or overfitting. The latter case is achieved by training the model to an arbitrary

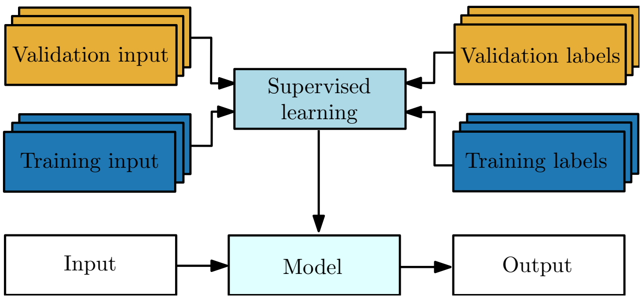


Figure 2.7: Generic training framework in a supervised learning setting.

number of epochs large enough so that the model achieves overfitting but then saving the value of θ that achieves the minimum validation loss. The former is achieved by saving the values of $\mathbf{\Theta}$ for which the model reaches the best validation loss. Finally, the test set is used as representative of deployment conditions to characterize the model's performance. For this purpose, the test set is kept isolated during training and only used in inference. The outcome of the network's performance on the test set should not cause a feedback loop to change $\mathbf{\Theta}$, as this could introduce a more subtle form of overfitting, inducing the design to believe the model is robust for deployment in real conditions while it has simply overfitted.

To train NN, gradient-descent methods based on backpropagation are the primary method to optimize θ. However, alternative approaches that address its drawbacks are being actively investigated in the literature. This training strategy is computationally intensive since it relies on multiple iterations and requires two data passages over the network. An alternative direct approach that avoids the backward pass using a regularized least-square method is presented in Section 2.2.3.

As with any data-driven model, it is important to stress that the final performance is driven by the quality and capability of the data to represent the real-world conditions in which a model will be deployed. If the difference between real and test data is too big, then a domain gap exists between the data used to train and assess network performance and the environment of the real world. If the data is corrupted or wrong, a model taking garbage as input will produce garbage as output.

2.2.1.6 Regularization

Regularization is a set of techniques used to prevent overfitting and improve the generalization performance of a model. The simplest form of regularization is achieved by including additional terms in the loss function used during training:

$$L = L_D + L_W \tag{2.13}$$

$$L_W = \sum_k \|\mathbf{w}_k\|^p \tag{2.14}$$

where L_D represents the loss component due to the data, and L_W defines the regularization term on the weights. Quadratic or p-norm penalty terms can improve the system's conditioning and reduce overfit[22]. To make large weights smaller, it is possible to set $p = 2$, resulting in L_2 regularization, while setting $p = 1$ can drive some of the weights to zero, resulting in L_1 regularization, also called Least Absolute Shrinkage and Selection Operator (LASSO).

Another form of regularization consists of data augmentation: increasing the number of training samples by properly perturbing the ones already available. Data augmentation is widely used for datasets for which it is expensive to generate labeled samples. Considering images, data augmentation consists of stretching, rotations, noise addition, cropping, resizing, and flipping, which can change the image's appearance. While this is trivial for classification tasks (since the labeled class associated with the image remains unchanged by the modification over the input), it requires particular care for regression tasks, as illustrated in Section 3.1.1.5.

Dropout is a powerful regularization technique introduced in [77] based on the random clamping of the neurons within a layer. By assigning a dropout percentage to one or multiple layers during each iteration of the training, random neurons corresponding to that percentage are disabled. Dropout prevents the networks from specializing specific units to certain tasks or samples, significantly improving generalization. Dropout is also a form of ensembling since it enables multiple architectures to live within the same model by exploring randomly activated neural paths. The rates and layers in which this strategy is implemented can be considered a model's hyperparameters, thus needing careful tuning.

2.2.1.7 Transfer learning

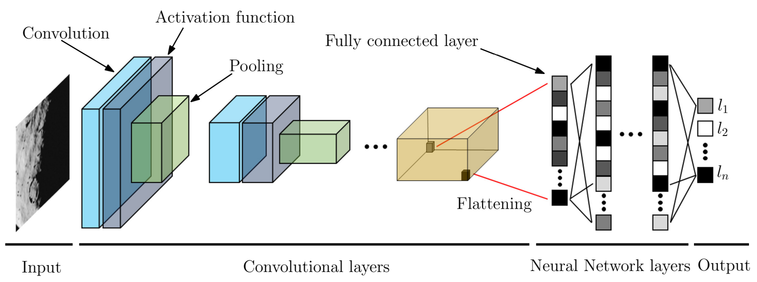
Transfer learning is a powerful technique that transfers knowledge learned from a task by re-using it to boost performance in learning a new and related task. Since training models, especially deep ones, may require substantial computational power, an established approach is to use transfer learning to get a fully trained model or a portion of it. This is then considered for the initial distribution of weights and biases and is fine-tuned via a new training phase specifically for the task considered. The fine-tuning may consider only a portion of the model or its entirety, freezing the weights and biases that shall not be changed during the training for the new task.

2.2.2 Convolutional Neural Networks

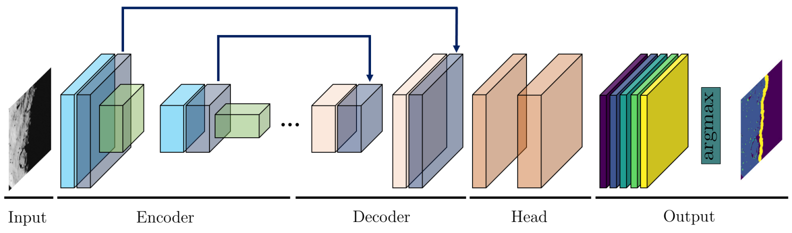
CNN have been first introduced as a succesful application in [78] for digit recognition and then popularized for image classification in [79]. Since then, they established themself as the principal state-of-the-art approach for image analysis.

Compared to traditional NNs, CNNs are characterized by deep architectures excelling in correlating spatial information. The architecture of a CNN augments that of traditional NN with specific blocks developed for tensorial operations. The main difference between conventional NNs and a CNNs lies in the fact that instead of connecting all the units in a layer to all the units in its preceding layer, CNN organize each layer into feature maps while conserving spatial information.

Typical convolutional architectures are schematized in Figure 2.8. The most used ones in this manuscript are hierarchical convolutional architectures (for regression and classification, exemplified by the architecture in Figure 2.8(a)) and UNet architectures (for segmentation, exemplified by the architecture in Figure 2.8(b)). Variants of these architectures are introduced and explained when needed throughout the manuscript. In contrast, this section describes the most important properties of CNN architectures with a standardized notation.



(a) CNN for regression or classification.



(b) CNN for image segmentation.

Figure 2.8: Schematic of CNN architectures used in this manuscript for regression or classification (a) and image segmentation (b).

Generically, a typical CNN architecture can be divided into different elements depending on the architecture considered, as displayed in Figure 2.8:

- **Input**: The input $\mathbf{X}$ represent the $w \times h \times d$ tensor representing the batch $\mathcal{B}$ processed by the network.
- **Encoder**: A sequence of image processing operations (generally represented by convolutional, pooling layers, and activation functions) is applied to extract and correlate spatial information while reshaping the input tensor, decreasing its height and width, and increasing its depth. This network portion encodes the data into a smaller but richer volume. In a classification/regression type of architecture, the encoder is represented by the convolutional layers up to the flattening operations that generate the fully connected layer. This is similar to a segmentation network based on the UNet architecture, with the main difference being the lack of the flattening operation.

- **Fully connected layer**: in a CNN for regression or classification, the convolutional layers need to be interfaced at one point with a traditional NN portion of the network. A reshape operation is thus required to flatten a 3D tensor into a 1D vector referred to as the fully connected layer, which constitutes the input layer for the NN in the head of the architecture.
- **Decoder**: In a hierarchical convolutional architecture, the decoder uses the highly abstracted representation generated by the encoder to generate a prediction. This representation is said to be in the latent space and essentially represents the data in a more meaningful and reduced form. A NN (either shallow or deep) is implemented to map the connection between the neurons from the fully connected layer, which embeds spatial information extracted from the image in the latent space, and the output layer, which expresses the desired output label of the entire architecture. In a hierarchical architecture, the decoder and the head often coincide. In a UNet architecture, the decoder comprises the inverse processing steps of the decoder concatenated with information from the encoder via skip connections.
- **Skip connections**: These are connected tensors that flow from the encoder to the decoder. These connections are peculiar to UNet architectures and are used to retain spatial information.
- **Head**: This portion is the final section of the network. In transfer learning, this is often the only processing portion of a model whose weights and biases are changed via fine-tuning. In hierarchical convolution architecture, the head is a portion or the entirety of the neural network layers, which are considered the decoder section of the network.
- **Output**: The output $\mathbf{Y}$ of the network. Depending on the task, this could be a tensor, a vector of class probabilities, or a vector of regressed values.

The building blocks of CNN are briefly described hereafter. These blocks augment those of NN and are tailored to CNN architectures.

2.2.2.1 Convolution

Convolution is a mathematical operation at the core of the functioning of a CNN. Given a generic tensor I (which could represent an image) and a smaller matrix K, referred to as "kernel", the 2D convolution operation between the two can be performed as:

$$C(x,y) = I * K = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(x-i, y-j) \cdot K(i,j) \tag{2.15}$$

where x, y are spatial coordinates over the input image, and $C(x,y)$ is the output of the convolution, also referred to "feature map" or "activation map". The feature map is obtained during the convolution by sliding the kernel over the entire input.

Convolution is a classical image-processing operation. Using different kernels, it is possible to obtain feature maps excited by a particular kernel structure. This is often called filtering within a traditional IP context. When the components of the kernels are explicitly defined, they can be used to highlight edge, blur, or sharpen the input image, as illustrated in Figure 2.9 using the following kernels:

$$\text{Identity} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.16}$$

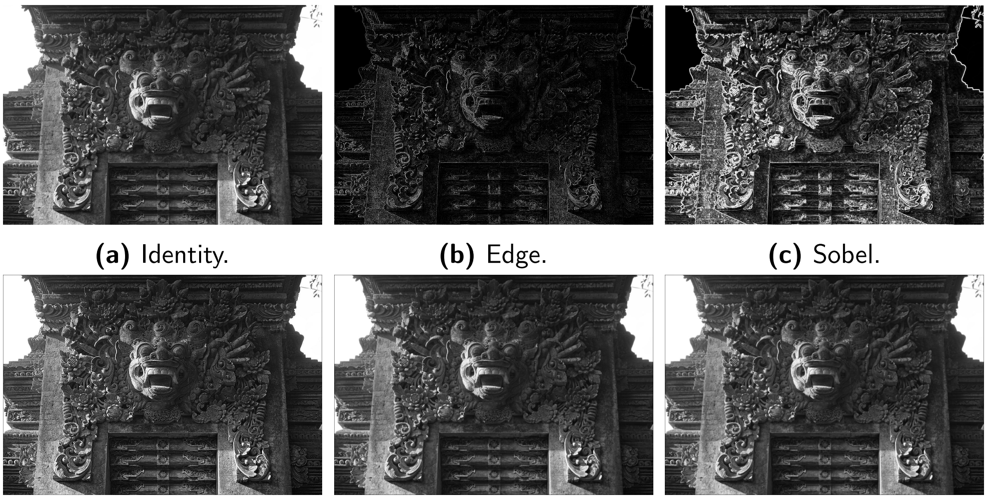
$$\text{Edge detection} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{2.17}$$

$$\text{Sobel opertor} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \tag{2.18}$$

$$\text{Sharpen} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{2.19}$$

$$\text{Gaussian blur} = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{2.20}$$

$$\text{Box blur} = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.21}$$



(a) Identity. **(b)** Edge. **(c)** Sobel.

(d) Sharpen. **(e)** Gaussian. **(f)** Box.

Figure 2.9: Feature maps obtained by the application of explicit kernels over the representation of a Bhoma head guarding the top of the portal to the Tirta Empul temple in Bali.

In a CNN, however, the components of the kernels are not defined a-priori and are considered as weights and biases of the network. By doing so, a great variety of implicit feature-extracting filters can be designed and combined in the hierarchical structure of the CNN, applied to images, and then recursively to their feature maps. Convolution operations are parametrized by padding, stride, and dilation.

Padding refers to the amount of pixels added to the input when the kernel processes it. This is relevant when a kernel is applied over the edge of the input, and the missing pixels outside the input must be specified for the convolution to occur. The stride represents the movement in terms of the number of pixels that the kernel performs from one convolution to the next. Finally, the dilation rate expands the kernel by inserting holes between its consecutive elements. In essence, it represents a convolution with pixel skipping to cover a larger input area. Dilated convolutions pool over a larger input region using fewer operations and learnable parameters. Examples of dilated convolutions are illustrated in the architecture in Figure 4.18 in Section 4.2.2.

The inverse operation of convolution is called deconvolution and is used in the decoder of segmentation architectures in a similar way in which convolution is used.

2.2.2.2 Pooling

Pooling is an operation to downsample the activation maps generated by convolutions. The most commonly used pooling forms are max and average, in which the maximum or average values from a specified region are passed to the subsequent layers. Pooling is essential in the hierarchical structure of CNN to synthesize spatial information, decreasing the data flow in the deep portion of the network.

2.2.2.3 Flattening

Flattening is a simple reshaping operation that transforms a three-dimensional tensor into a one-dimensional vector. The primary purpose of flattening is to unfold the last feature map generated by the convolutional layers into a dense fully connected layer. This format can be used to interface with traditional NNs.

2.2.3 Convolutional Extreme Learning Machine

An alternative approach to deep architectures and learning via GD exists, which in some cases has been demonstrated to perform similarly or better. In this section, an overview of this approach is presented.

2.2.3.1 Introduction to extreme learning

Extreme Learning Machine (ELM) is a theoretical formulation of a learning strategy first introduced about two decades ago in [80] and later organized more consistently first in [81] and then in [82]. In these works, ELM theory is applied to single-layer feedforward networks whose weights and biases are randomly initialized. Training uses a Leas Square (LS) method to adjust only the weights of the connections between a single hidden layer and the output one.

Training happens extremely fast because LS is an order of magnitude faster than GD. With enough randomized neurons in the hidden layers, a network could generate a multi-dimensional basis that can be used to map the nonlinear relationship between input and labels. ELM is demonstrated to perform similarly or better than deep architectures [80–82], requiring only a fraction of their training time.

At the same time, ELM concepts were being formalized and used, the pivotal work in [83] stressed the unexpected performance achieved with CNN when using random weights and biases in the convolutional kernels considering generic computer vision tasks. The authors in [83] prove that:

1. A surprising fraction of performance in a CNN can be contributed by the intrinsic properties of the architecture alone and not from the learning algorithm used (these include settings such as the number of layers, the depth of the kernels, and the activation functions used).
2. Convolutional pooling architectures can be frequency selective and translation invariant, even when random weights are used.
3. A methodology that uses randomized CNN to search the hyperparameters within the architecture design space performs inherently better than traditional approaches. An order of magnitude speedup in the training process is obtained by sidestepping the time-consuming learning process and only focusing on those architectures with superior hierarchical structures.

These two research lines come together in [84], which extends the ELM theory to CNN with randomized kernels, introducing the concept of Convolutional Extreme Learning Machine (CELM) for generic computer vision tasks. The convolutional layers of a CNN are set with random weights and biases up to the fully connected layer, whose connections with the output layer are treated as an ELM architecture and solved with a LS method. Similarly to ELM, CELM achieves extremely fast training and accuracies that may be similar to those of CNN. Since then, several other works using CELM architectures and training strategies have been presented in the literature over the past two decades, as summarized by the thorough and systematic review in [85].

Interestingly, from [85] and to the best of the author's knowledge, no prior work has been focused in the past two decades on the adoption of CELM for onboard IP applications that target celestial bodies. Albeit their application could seem outdated, they may be well suited when considering the typical scenery imaged by a navigation camera in the proximity of a small body, which is relatively simple when compared to other computer vision domains. The background and foreground are distinguishable, and the surface variations are only due to morphological characteristics (i.e., craters, boulders, etc.), which only vary under illumination conditions. This domain is fundamentally simpler than typical computer vision applications in urban environments, which need to account for many commonly used objects, where deep architectures are the state of the art [22]. Moreover, previous findings in [32] already hinted that the filtering capabilities of CNNs on images of a small body is the critical element that pushes the performance compared to traditional methods.

2.2.3.2 Training

In a traditional CNN architecture, convolutional layers are driven by weights and biases that only influence the kernels used in the convolution operations. On the other hand, the weights and biases of the neural network portion of the architecture represent the influence of the neuron connections between the fully connected and output layers.

For simplicity, when considering CELM, it is important to consider these two sets of weight and biases separately, as illustrated in Figure 2.10, since depending on the training strategy used within this manuscript, they will be handled differently. The former will be referred to as **W** and **b**, the latter to β and β_0. The set of weights and biases of the entire architecture defines the set of parameters referred to as $\theta = (\mathbf{W}, \mathbf{b}, \beta, \beta_{\mathbf{0}})$.

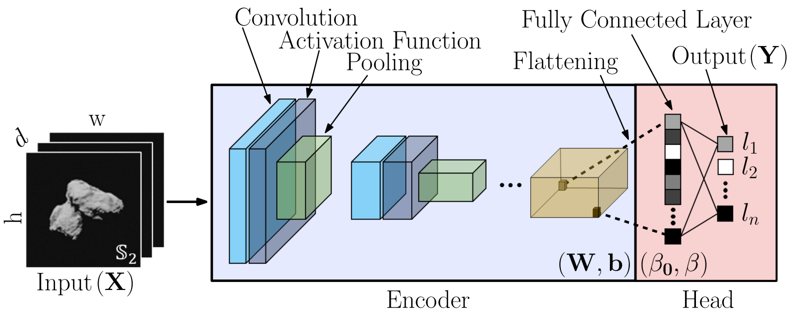


Figure 2.10: Division between the convolutional and neural network layers in a model trained using the CELM paradigm.

Apart from the weight distinction, from an architecture point of view, CELM and CNN can share identical structures. Indeed, the term CELM refers to the training strategy other than the architecture, but it is often convenient to refer to a CNN that has been trained using ELM theory directly as a CELM.

In a CELM, once the weights and biases of the convolutional layers are randomly set at initialization, they are considered frozen during training. On the contrary, β , are tuned during training. Given a set of true input-output samples, $(\mathbf{X}, \mathbf{T})$, the forward pass of the input into the network generates a hidden layer output matrix H right before the output layer:

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} h_1(\mathbf{x}_1) & \dots & h_L(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ h_1(\mathbf{x}_N) & \dots & h_L(\mathbf{x}_N) \end{bmatrix} \tag{2.22}$$

where L is the hidden layer's size before the output one, and N is the number of neurons in the output layer. The training data target matrix is then defined as:

$$\mathbf{T} = \begin{bmatrix} \mathbf{t}_1 \\ \vdots \\ \mathbf{t}_N \end{bmatrix} = \begin{bmatrix} t_{11} & \dots & t_{1m} \\ \vdots & \ddots & \vdots \\ t_{N1} & \dots & t_{Nm} \end{bmatrix} \tag{2.23}$$

To find the best set of weights β that matches the matrix $\mathbf{T}$, the following optimization problem shall be solved [84]:

$$\text{Minimize}: \|\beta\|_2^2 + C\,\|\mathbf{H}\beta - \mathbf{T}\|_2^2 \tag{2.24}$$

that is a regularized least square that depends on the C coefficient. Including the first term regarding the minimization of the weights vector β increases the stability and improves generalization capabilities [81]. The minimization problem can be solved efficiently by:

$$\beta = \begin{cases} \mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1} \mathbf{T}, & \text{if } N \leq L \\ \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1} \mathbf{H}^T\mathbf{T}, & \text{if } N > L \end{cases} \tag{2.25}$$

Since the remaining weights and biases ($\mathbf{W}$, $\mathbf{b}$, and β_0), are randomly fixed at initialization and are never changed, there is no need for a backward pass. Only the forward pass and the LS problem must be processed, making the training much faster than gradient-based methods.

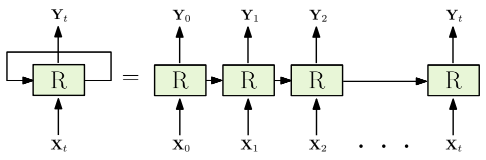
This advantage does not impact the computational effort at inference compared to the one of CNNs. It has to be intended instead as a tool that can be exploited to explore the design space of an architecture before the internal optimization of its weights and biases. This exploration, which is usually performed via search-grid methods [22, 85], can instead be performed more extensively thanks to the exceptionally fast training time of CELM.

2.2.4 Recurrent Neural Networks

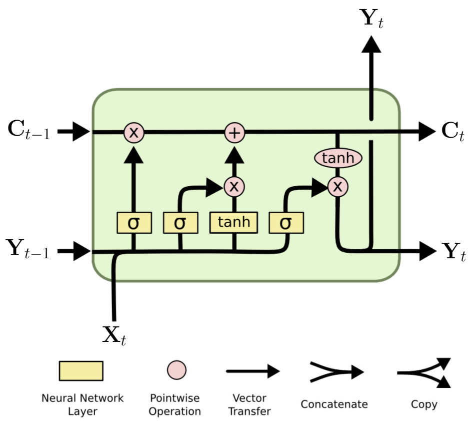
While the networks presented in the previous sections are specifically designed to operate with single feature vectors or images, producing pinpoint estimates, sometimes it is also helpful to process image sequences, fusing spatial information and temporal dependencies. For these types of tasks, the most suited form of architecture is represented by Recurrent Neural Network (RNN) [86].

Using multiple stages, data is processed sequentially in a RNN so that one stage's output feeds the following input. Recurrent connections are established between epochs, allowing information to persist over time. Although the weights are shared between all stages, each maintains its state and backpropagates its gradients.

Long-Short Term Memory (LSTM) are a particular family of RNN first introduced in [87] and then popularized in many other works. The design of LSTM is specifically thought to avoid the long-term dependency problem typical of traditional RNN, which performs poorly when a large gap exists between the relevant information and the point where it is needed. All RNN (when unfolded) can be represented by a chain of repeating modules, as illustrated in Figure 2.11(a). When a simple RNN is used, this could be populated by a straightforward structure, or in the case of LSTM, by a complex architecture composed of several gates that control the flow of information, as illustrated in Figure 2.11.



(a) Unrolled RNN architecture.



(b) Typical LSTM structure, from [10].

Figure 2.11: An unrolled RNN (a) and a zoom over a LSTM unit (b).

As depicted in Figure 2.11(b), the flow of information is controlled by three gates, represented by σ layers. Note that the cell state C flows almost unperturbed between all cells, ultimately allowing the LSTMs to perform better than traditional NNs.

As LSTM are exceptional in image-sequence analyses, they are used in this research to improve position estimates and to generate velocity estimates from position sequences.

[10] https://colah.github.io/posts/2015-08-Understanding-LSTMs/, last accessed 31st of August 2023.

3

Data generation

"I put a spatially tessellated void inside a modified temporal field until a planet developed intelligent life. I then introduced that life to the wonders of electricity, which they now generate on a global scale. And, you know, some of it goes to power my engine and charge my phone and stuff."

Rick Sanchez, *S2E6*

Access to high-quality data-label pairs is fundamental for data-driven algorithms trained with supervised learning methods. Unfortunately, due to the limited imaging of celestial bodies throughout the history of space exploration, there is a practical constraint regarding the availability of such data for computer vision applications in this field. This limitation adversely affects the ability to create and train data-driven algorithms and the capacity to validate their functionality before deployment.

This chapter delves into this issue by showcasing three approaches to address this challenge with increasing complexity, cost, and fidelity. Firstly, an artificial environment can generate synthetic yet realistic samples. In this approach, one has complete control over the environment, allowing for the simple generation of labels. This represents the simplest form of data generation, with the primary advantage being its accessibility to a wide range of users due to its cost-effectiveness. However, it requires a highly accurate modeling of the environment to be effective. While there is no constraint on the volume of data generated for training, validation is complex but can be reached by comparing the model's performance with real images from previously flown missions.

Another alternative involves using an optical facility equipped with a representative camera. This setup is helpful to validate the IP, since including a real camera enables the simulation of a subset of typical errors and enhances the realism of the environment. This approach also allows for the execution of algorithms on a representative processor connected to the camera interface.

Lastly, a physical facility provides the most authentic setting, resembling the actual environment. However, using 3D shape models in robotic facilities or analog terrain models can be rigid and costly, often seen as the final stage in an incremental validation campaign. Data acquired in this setting is precious but rarely shared.

3.1 Synthetic rendering

The Celestial Object Rendering TOol (CORTO) is a tool under development since Summer 2020 at the Deep-space Astrodynamics Research & Technology (DART) group[11] at Politecnico di Milano. The tool's objective is to enable the high-fidelity, flexible, and simple generation of artificial image-label pairs of celestial bodies that can be used both to design and test IP and visual-based navigation algorithms. The motivations behind the tool development are multiple and address pressing issues faced by the authors of this work in research and industrial contexts.

First, open datasets of real imagery of close-up views of celestial bodies such as planets, moons, asteroids, and comets are scarce. This is due to a combination of different limiting factors: 1) We currently do not have an extensive sample of visited bodies in the Solar System, especially concerning asteroids and comets.; 2) The existing datasets are limited by the mission geometry (e.g., in the case of flybys, often only a single face of the body is imaged at high resolution), posing stringent viewing and illumination conditions of the existing images; 3) Not all datasets from previous missions are publicly available (different space agencies have different dissemination strategies) and even when they are released for use by the broader engineering community, they are often not released shortly after arrival due to embargo reasons and priority given to scientific investigations, introducing a delay into the availability of the images by researchers interested into visual-based applications.

Second, to design, validate, and test image processing algorithms that use celestial bodies as targets, a common approach is to generate and use comprehensive high-fidelity datasets. Due to the lack of control over the viewing and illumination conditions of real-mission images, creating a digital model of the target body is often preferred to explore the search space without limitations, both in terms of design and testing of the algorithm. An established approach is thus to artificially simulate the visual environment with the fidelity required by the algorithm to complement the existing data. This can be performed in two ways. When synthetic datasets are made out for existing and well-known bodies, they can augment existing images with additional geometric and illumination conditions. On the other hand, considering bodies that have never been imaged, synthetic datasets can be used to realistically represent them, providing a powerful tool for mission designers.

Third, as an increasing number of image-processing algorithms are being explored with data-driven design, a critical drawback exists for their adoption for space applications. These algorithms are often data-hungry, so their development clashes with the lack of data that characterizes space applications, particularly minor bodies. Moreover, supervised methods do not simply need data but also associated

[11] https://dart.polimi.it/, last accessed 8th of August, 2023.

labels. For example, reconstructed positioning data from a real mission can be used to represent the flew trajectory with an estimation error and used as a label for navigation algorithms. However, applications such as semantic segmentation may require data (e.g., a pixel-per-pixel classification of the classes of morphological features of an image) that is far more complex to generate.

Fourth, at the time of development of CORTO, tools capable of artificially recreating the environment of celestial bodies were neither open-source nor easily accessible by individual researchers. Given the capabilities these tools unlock, their access is often regulated via licensing or is generally kept confidential since they provide a strategic advance. Moreover, their development requires a substantial investment of time and competencies that may not match that of an image-processing designer. Consequently, there has been a proliferation of different useful tools, which has created the effect of isolating communities and avoiding sharing datasets and tools for image generation. Among the most notable ones:

- Planetary Planet and Asteroid Natural scene Generation Utility (PANGU)[88–90] is considered the state-of-the-art for rendering celestial bodies. It is a tool with robust, long-lasting, and documented development designed by the University of Dundee for the ESA. PANGU supports various advanced functionalities and is extensively used as the industry standard for ESA projects involving visual-based navigation algorithms. However, access to the software is regulated via licenses and often requires direct involvement with an ESA project as a pre-requisite.
- SurRender [91] is proprietary software by Airbus Defense and Space[12] that has been successfully used in designing and validating various vision-based applications for space missions in which the company is involved. The software can handle objects such as planets, asteroids, stars, satellites, and spacecraft. It provides detailed models of sensors (cameras, LiDAR) with validated radiometric and geometric models (global or rolling shutter, pupil size, gains, variable point spread function, noises, etc.). The renderings are based on real-time image generation in OpenGL or raytracing for real-time testing of onboard software. Surface properties are tailored with user-specified reflectance models, textures, and normal maps. The addition of procedural details such as fractal albedos, multi-scale elevation structures, 3D models, and distributions of craters and boulders are also supported.
- Space Imaging Simulator for Proximity Operations (SISPO)[92] is an open-access image generation tool developed by a group of researchers from the universities of Tartu and Aalto, specifically designed to support a proposed multi-asteroid tour mission [93] and the ESA's Comet Interceptor mission [94]. SISPO can obtain photo-realistic images of minor bodies and planetary surfaces using Blender[13] Cycles and OpenGL[14] as rendering engines.

[12]https://www.airbus.com/en/products-services/space/customer-services/surrendersoftware, last accessed 8th of August, 2023.

[13]https://www.blender.org/, last accessed 8th of August, 2023.

[14]https://www.opengl.org/, last accessed 8th of August, 2023.

Additionally, SISPO makes available advanced scattering functions written in Open Shading Language (OSL) that can be used in the shading tab in Blender to model surface reflectance, greatly enhancing the output quality.
- The simulation tools illustrated in [95, 96], that implements high-fidelity regolith-specific reflectance models using Blender and Unreal Engine 5[15]. The tools can render high-fidelity imagery for close proximity applications, particularly about small bodies, focusing on the high-fidelity simulation of boulder fields over their surfaces.
- AstroSym[97], developed in Python to provide a source of images for closed-loop simulation for Guidance Navigation and Control (GNC) systems for landing and close-proximity operations around asteroids.
- SPyRender[44], also developed in Python, is used to generate high-fidelity images of the comet 67P for training data-driven IP methods for navigation applications.

These challenges motivated the need to develop a simple-to-use, easy-to-learn tool to enable researchers and students to generate realistic images to train or test their image processing algorithms. This effort resulted in the development of CORTO, which has primarily targeted applications around small bodies and the Moon.

The tool is designed with a modular structure using various software and libraries. The realism of the final images is intended to reduce the domain gap between synthetic and real images, providing a powerful tool to a mission designer. This section describes the main functionalities of CORTO in detail. Great care is also put into showing the validation of the tool's output and the applications in which CORTO is currently being used.

3.1.1 Architecture

CORTO is at the core of the tools developed and used to simulate the visual environment around celestial bodies. CORTO can be used to generate image-label pairs both online (with GNC systems connected in closed-loop) or offline (to create datasets for statistical analysis or design). The tool's core is the Blender software, an open-access rendering software that is easy to learn, has a large community, and supports Python scripting. The main functionalities are handled in Python, while Matlab, Simulink, and SPICE are used within the tool.

The general architecture of CORTO, as well as its relationship with other tools and simulators of the DART group, is illustrated in Figure 3.1. The inputs to CORTO are *Scene*, *Geometry*, and *Body* configuration files, while the outputs of the tool are images (I) and labels (L). The tool can be used in open-loop by using a set of pre-defined configuration files containing all the setup to be used for all the renderings or in closed-loop by updating only the *Scene* and *Geometry* inputs after the generation and processing of each image. Currently, the tool is not designed

[15]https://www.unrealengine.com/en-US, last accessed 8th of August, 2023.

for real-time rendering but focuses on dataset generation and delayed closed-loop simulations.

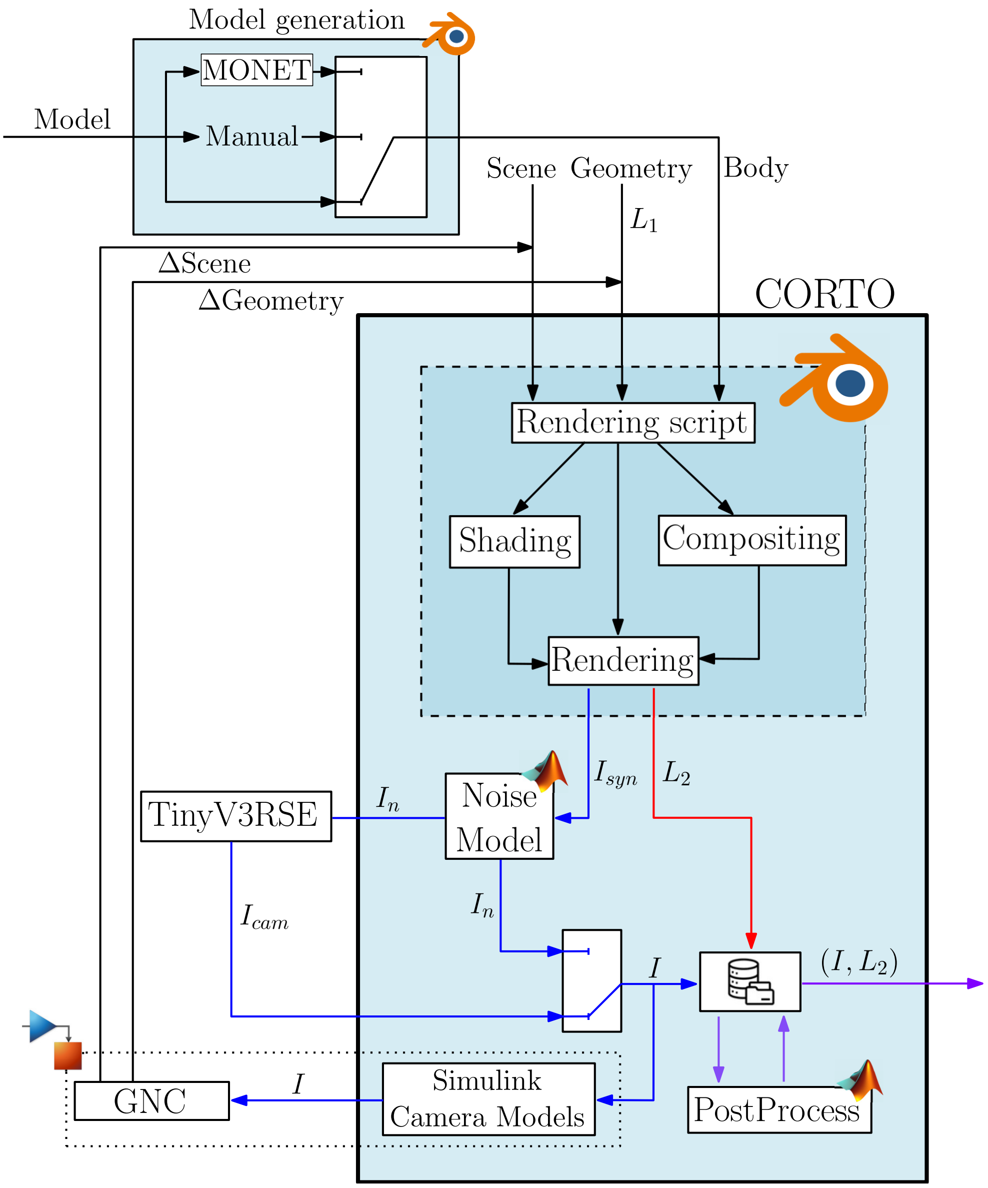


Figure 3.1: High-level architecture of CORTO and other tools used at the DART lab.

A critical functionality is the capability to generate labels associated with the images. These are divided into two groups: L_1 and L_2. L_1 represent quantities also used as input in Blender to position the objects in the scene (e.g., the poses of all the bodies and derivatives quantities such as the range, phase angle, and others). L_2 represent labels generated in Blender during the rendering process (e.g., pixel classes, depth maps, slope maps, and others). Both sets of labels can be used and coupled with images, but it is essential to distinguish between them, as they are of a different nature.

Below, the flow of how to use the tool is now exemplified, while the core blocks are explained in detail in the following subsections. The starting point is the *Model generation* block in Figure 3.1, which takes as input a rough mesh model saved as a ".obj" and generates the *Body* input for CORTO. This block can be used to augment an existing object into a high-resolution one with morphological features of interest.

To operate, CORTO needs other inputs, referred to as *Geometry*, and *Scene*. The *Geometry* input consists of a configuration file in which the poses of the objects involved in the scene are handled. These include the spacecraft position (r_{SC}) and orientation (q_{SC}), the body position (r_B) and orientation (q_B), and the Sun orientation (q_S) all in a shared reference system, as it is possible to see in Figure 3.2. Finally, the *Scene* configuration file contains data about the camera, material, and illumination properties of the Sun's lamp.

The *Body*, *Geometry*, and *Scene* inputs are then read by a Python rendering script. The script is the core component of CORTO and is used to set up all necessary configurations for the *shading*, *compositing*, and *rendering* tabs in Blender. The script manages the renderings and the generation of the image-label pairs, denoted as I_{syn} and L_2. An example of scene rendering from CORTO input is illustrated in Figure 3.2. Note that the Sun's geometrical settings are commanded only by its orientation q_S and are invariant to its position. For simplicity, it can be conveniently fixed to the center of the target body.

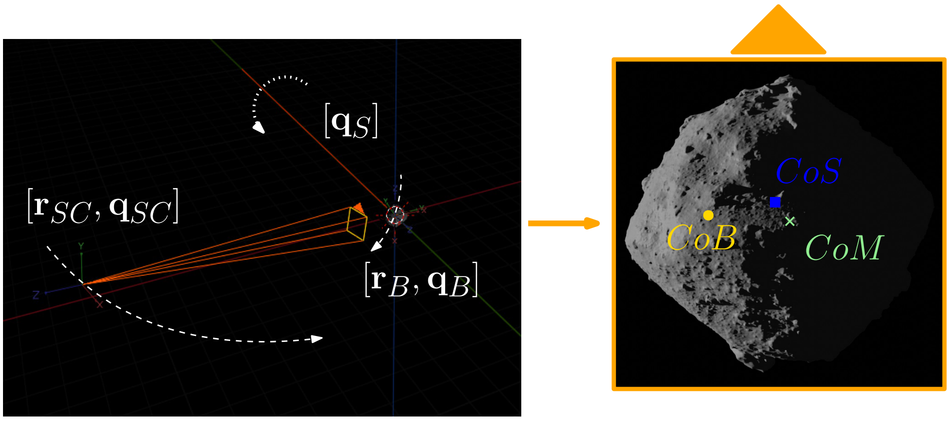

$[\mathbf{q}_S]$
$[\mathbf{r}_{SC}, \mathbf{q}_{SC}]$
$[\mathbf{r}_B, \mathbf{q}_B]$
CoS
CoB
CoM

Figure 3.2: Example of scene generation from *Geometry* input. Position and orientation of the objects in the scene in Blender (left) and generated image (right).

After generation, images can pass into the *Noise model* block implemented in Matlab, which adds artificial noise to the synthetic images I_{syn}. The noise can represent camera and environmental disturbances, and its addition can be selectively turned off. This is particularly relevant when performing Hardware-In-The-Loop (HIL) simulations since the camera noise can be modeled directly by the stimulated camera. For this purpose, the Tiny Versatile 3dimensional reality simulation environment (TinyV3RSE) facility [98, 99] can be used to include a camera within the loop of the IP algorithm and to generate noisy images with it.

The output image I after noise addition (either artificially generated or with an HIL setup) can then follow two routes depending on the open-loop or closed-loop settings. In an open-loop scenario, the image is saved in a database. In a closed-loop scenario, the images are transferred to a module in Simulink that acts as a virtual camera sensor that stimulates a complete GNC subsystem. This approach uses TCP/IP protocols to transmit images and flag commands that generate the subsequent spacecraft pose, repeating the entire cycle.

Finally, a postprocessing step can be applied to prepare the image-label pairs to constitute a dataset. This is especially relevant for data-driven algorithms (that may require data augmentation, cropping, and resizing) but also applies to traditional algorithms.

Domain randomization is a powerful technique that allows for a large variation of the possible image space. In this work, this is intended in terms of the appearance of the input image. Properly tuning the settings within the rendering, noise modeling, and postprocessing blocks allows to obtain geometrically identical samples of the same image with randomized appearance. This technique can be used as a data-augmentation technique to enable a data-driven model to become robust about noise, albedo, illumination, material properties, and body position within the input image.

As explained before, the core capability of CORTO is not only that of generating high-fidelity images but also of generating labels accompanying such images. What follows is a set of examples of labels that can be obtained from CORTO. In Figure 3.3, it is possible to see an image of the Didymos system and associated pixel labels that could be used for object recognition to distinguish between Didymos, the primary body, and Dimorphos, its secondary. These masks are obtained at rendering by properly setting the pass indices of the bodies.

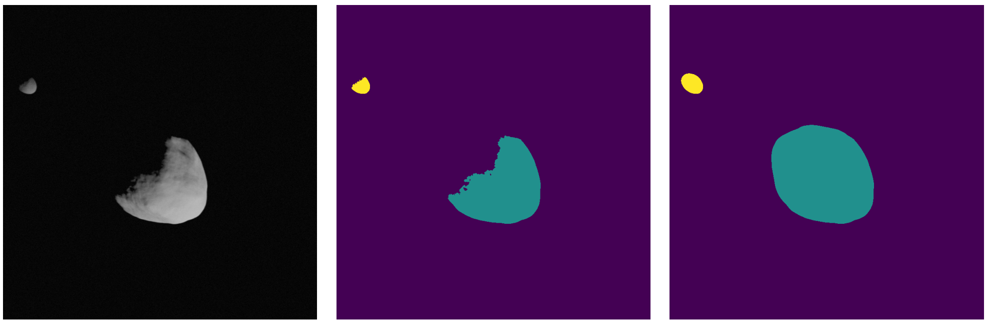
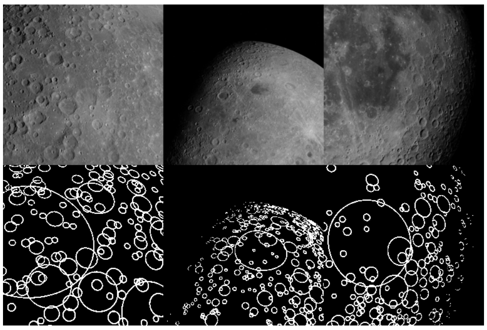

(a) Input grayscale image. **(b)** Masks w. shadows. **(c)** Masks w.o. shadows.

Figure 3.3: Image and associated masks for the background (purple), Didymos (green), and Dimorphos (Yellow) classes.

In Figure 3.4, it is possible to see different labels about craters on the Moon and on Ceres generated using high-fidelity texture maps obtained from the Robbins [100] crater dataset [16] and the Zeilnhofer crater dataset [17].


(a) Moon craters.

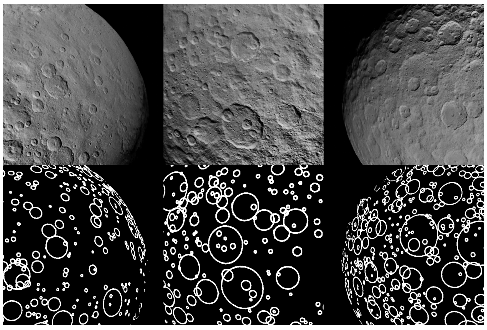

(b) Ceres craters.

Figure 3.4: Example of images with crater labels obtained in Blender by elaborating a high-fidelity texture map from the Robbins and Zeilnhofer datasets.

In Figure 3.5, examples of depth-map and slope labels are generated for asteroid Ryugu and comet 67P.

Finally, in Figure 3.6, it is possible to see different examples of segmentation maps that can be used generated about small bodies that exploit different pass indices strategies to obtain multi-layers maps, single class maps (boulders-surface) and hierarchical ones (background, surface, small boulders, and prominent boulders divided by single identifiable color codes).

3.1.1.1 Object handling

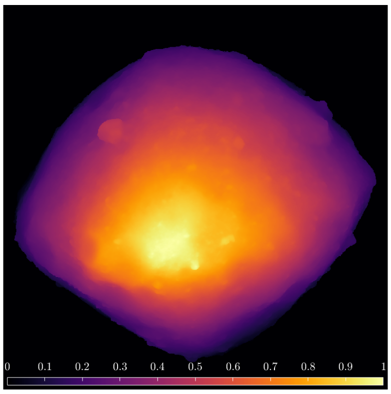
One of the most critical inputs to CORTO is the shape model of the target body. Depending on the observation technique used to generate it, available shape models can be rough (e.g., when observed with radio telescopes from Earth or during a flyby) or accurate (e.g., when observed during a rendezvous mission).

Each model passes through the *Model generation* block independently from the source, following one of three possible paths. If the model is already accurate for the task considered, it can be passed as it is directly as a ".obj" to CORTO. However, in many cases, the shape model is not sufficiently accurate for the task considered. In these cases, manual and procedural modifications are dedicated to refining the model mesh and introducing morphological features such as roughness, craters, and boulders over the surface. These are added in Blender, but in principle, any software capable of modifying a 3D mesh can be used.

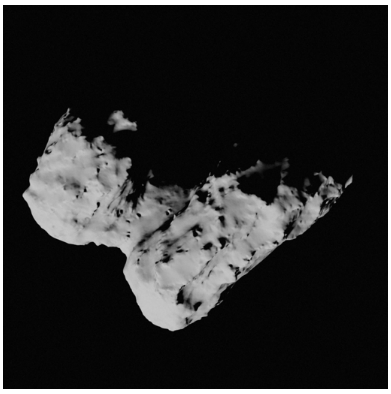
[16] https://astrogeology.usgs.gov/search/map/Moon/Research/Craters/lunar_crater_database_robbins_2018, last accessed 8th of August, 2023.

[17] https://astrogeology.usgs.gov/search/map/Ceres/Dawn/Craters/ceres_dawn_fc2_craterdatabase_zeilnhofer_2020_v2, last accessed 8th of August, 2023.

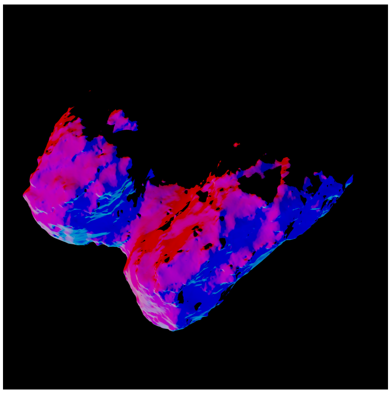
(a) Input image of Ryugu.



(b) Depth map label of Ryugu.



(c) Input image of 67P.



(d) Slopes label of 67P.

Figure 3.5: Examples of the depth map and slope labels with asteroid Ryugu and comet 67P.

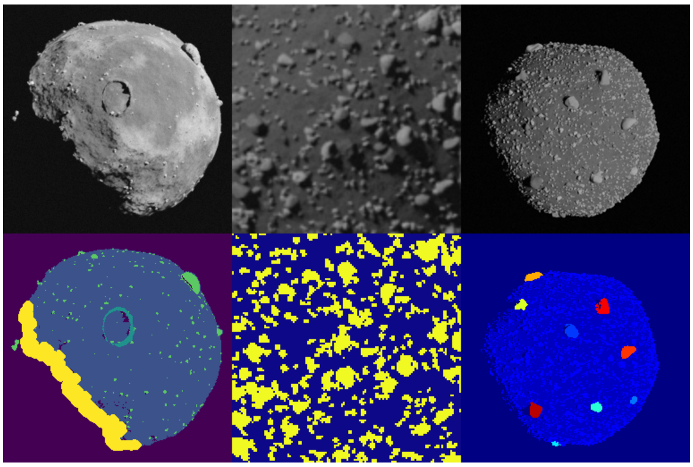


Figure 3.6: Examples of masks that can be generated for semantic segmentation applications.

When manual refinement is performed, details are arbitrarily added over the surface. For example, the mesh can be thickened, surface roughness can be achieved using noise elements as textures, and craters and boulders can be introduced using Blender's built-in tools. This allows objects to be placed in the desired positions and appearance. The entire process is manual and artistic, does not allow for reproducibility, and requires skills from the user in working with 3D objects [101].

Lastly, the model can be procedurally modified by using Minor bOdy geNErator Tool (MONET) [102] (previously also referred to Procedural Asteroid Generator or PAG) designed at DART. MONET takes the model as input, automatically refines and smoothes the mesh, and introduces morphological features such as roughness, craters, and boulders from user-defined input values. By default, MONET is capable of realizing two different categories of minor bodies, namely rocky bodies (characterized by a large number of different-sized boulders on the surface) to simulate rubble-pile asteroids and comet-like bodies (whose surface exhibits the alternation of very smooth and rough regions typical of comets). Adjusting the tool's settings makes it possible to model the properties of various minor bodies in a complete procedural approach. The way in which roughness, craters, and boulders are added to a body is briefly described. A more detailed explanation is reported in [102].

To generate a rough surface, a material is applied to the body's mesh, exploiting the node tree available in the *shading editor* in Blender. A *noise* texture is used into a *bump* node to displace the object's surface along the face's normal while breaking the uniformity of the surface's color. An example of surface roughness achieved with this simple approach is illustrated in Figure 3.7

(a) Noise texture. **(b)** Simulated roughness.

Figure 3.7: Surface roughness visualized on a model. Noise texture applied on the model (a). Displacement and bump on the model given by such texture (b).

To generate procedural craters, two different methods have been used. The first consisted of exploiting an existing crater's texture to develop its 3D model and then applying it on the surface of the minor body model thanks to a *shrinkwrap* modifier. This method is used in [103] but was best suited for smooth, flat surfaces and not over irregular terrain patches. The second method, representing the current baseline, uses a texture map based on the *voronoi* pattern.

Such patterns are added by a *color ramp* node with four different shades of black to obtain the typical effect of an excavation. From this point onward, the set of operations illustrated in Figure 3.8 is performed using the roughness texture.

Subtraction is performed between the roughness and the preliminary craters' texture to solve an issue regarding the extension of the surface roughness into the cratered region. This operation generates a hybrid roughness texture in which white circular areas delimit craters. This hybrid texture is then combined with the preliminary crater's texture using the *mix* node[18]. Finally, the *bump* node is exploited to modify the object's surface. This approach can be performed multiple times in the node tree starting from different sizes *voronoi* textures to generate overlapping craters of various sizes, positions, and distributions.

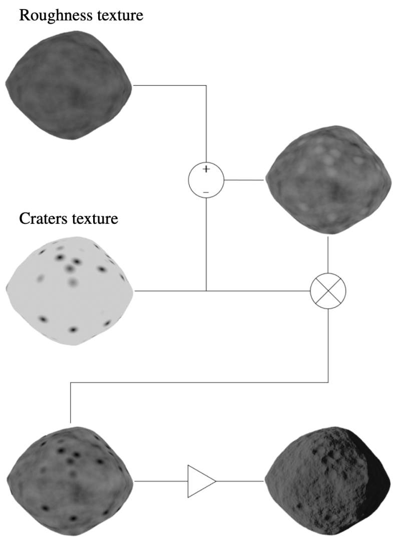
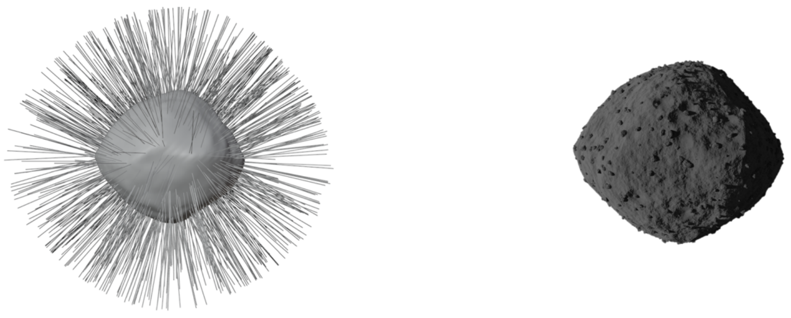


Figure 3.8: Craters generation procedure. The operations performed on the textures are schematically reported, along with the final result depicted on the bottom right. $\otimes$ represents the *mix* node and multiplication, while $\triangleright$ the *bump* node.

To distribute boulders over the body's surface, an approach is designed in Blender exploiting its native particle system and an add-on tool named *rock generator*[19]. First, a set of boulders is generated using the *rock generator*. The maximum number of samples generated is limited by the hardware capabilities to handle additional meshes. In contrast, the minimum number of samples is limited by the desired variability in the boulder's population. Exploiting the Blender's particle system (in particular, the portion simulating the "hairs" of an object), it is possible to sample the previous set of boulders and scatter them across the surface with randomized orientation, scales, and locations. Note that complex distributions can also be achieved by using different particle systems, assigning each to different types or scales of boulders. In the current baseline, three particle systems control small, medium, and large boulders. In Figure 3.9, examples of the particle system positioning of the boulders and their final appearance on the body can be seen.

[18]https://docs.blender.org/manual/en/latest/render/shader_nodes/color/mix.html, last Access August 20th, 2023

[19]https://github.com/versluis/Rock-Generator, last access August 20th, 2023



(a) Particle system. **(b)** Boulders distribution.

Figure 3.9: Boulders generation. Visualization of Blender's particle system (a) and final distribution of boulders (b).

Finally, the entire process to perform procedural changes with MONET is schematized in Figure 3.10

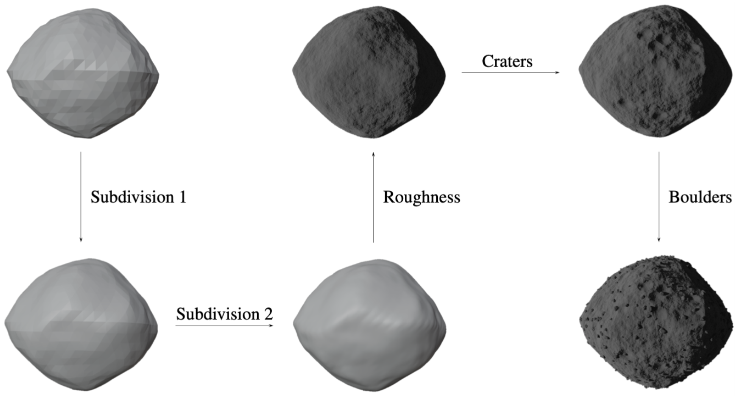


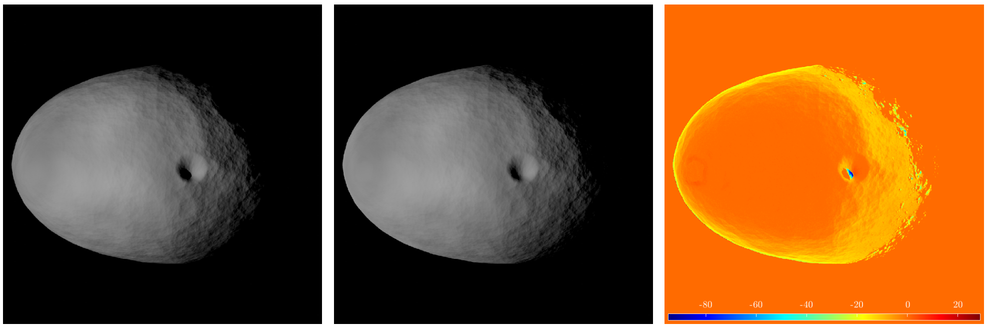
Figure 3.10: Full pipeline in MONET from input 3D model to final refined one.

3.1.1.2 Rendering

The rendering procedure is handled by a Python script but executed in Blender. Two different engines are used to generate the renderings: Cycles and Eevee. The former uses path-tracing algorithms and is considered more photo-realistic but resource-intensive. The latter uses simplified light environments and can render simpler scenes accurately in a shorter time. Considering the photorealism needed from the images, the type of labels desired, and the rendering speed with the available hardware, the user might prefer one of the two rendering engines.

In Figure 3.11, it is possible to see the rendering difference of the same scene of the Dimorphos asteroid when using Cycles and Eevee.

When building the body properties via the input parameters across the different shading, compositing, and settings tabs in Blender, it is essential to define the



(a) Cycles rendering. **(b)** Eevee rendering. **(c)** Rendering difference.

Figure 3.11: Example of images generated with different rendering engines in Blender.

requirements of the final image that best reflect the needs of the processing algorithm that will use such images. These will drive critical choices on the body properties and rendering settings to be used.

For example, the surface of a celestial body could be rendered using the standard Principled Bi-directional Scatter Distribution Falloff (PBSDF) implemented in Blender, combining it with a texture map, or employing ad-hoc scattering laws coded in OSL, as in [92]. Texture maps, however, are a scientific product of a mission and might not be available for the body of interest. Moreover, even if available, they could be generated accurately only for a portion of the body or with offset phase angles that may introduce spurious shadows into the images, invalidating their photorealism. At the same time, their high variability and close resemblance to the real surface albedo may be critical for a feature-tracking algorithm that may prefer them over a plain mesh surface modeled with a combination of scattering functions. Another alternative currently under investigation and based on recent works [104] focuses on assigning single scattering properties to each boulder scattered across the surface of a model of an artificial asteroid. This option, however, is at this stage of development computationally intensive and yet to be formalized in the nominal pipeline of CORTO. On the other hand, if the target application is a limb-based algorithm (or any other algorithm using global properties), the designer may neglect texture maps altogether and focus on the global appearance of the body and the simulation of the correct scattering function.

These considerations are crucial in designing an appropriate artificial environment for the task considered. The user might consider the pros and cons of each modeling strategy and select the one that best fits the algorithm-specific objectives and design. Currently, the surface of the object in CORTO can be simulated using the standard PBSDF, PBSDF in combination with texture maps, or a set of defined scattering functions introduced using OSL. Note that these functions are the same developed in [92] and are the: 1) Lommel-Seeliger, 2) ROLO, 3) Akimov, 4) Linear Akimov, 5) Lunar Lambert, and 6) Minnaert.

3.1.1.3 Noise Modelling

Adding noise into the synthetic images generated by CORTO is an important step to make them more similar to real images acquired by navigation sensors onboard previously flown missions.

The noise block in CORTO does that with the methodology summarized in Figure 3.12, which has been adapted specifically for visual-navigation space cameras from the noise modeling in [22] and in [105]. For this reason, the block is currently applicable to grayscale images only, and shutter or aperture effects are not considered assuming typical space cameras.

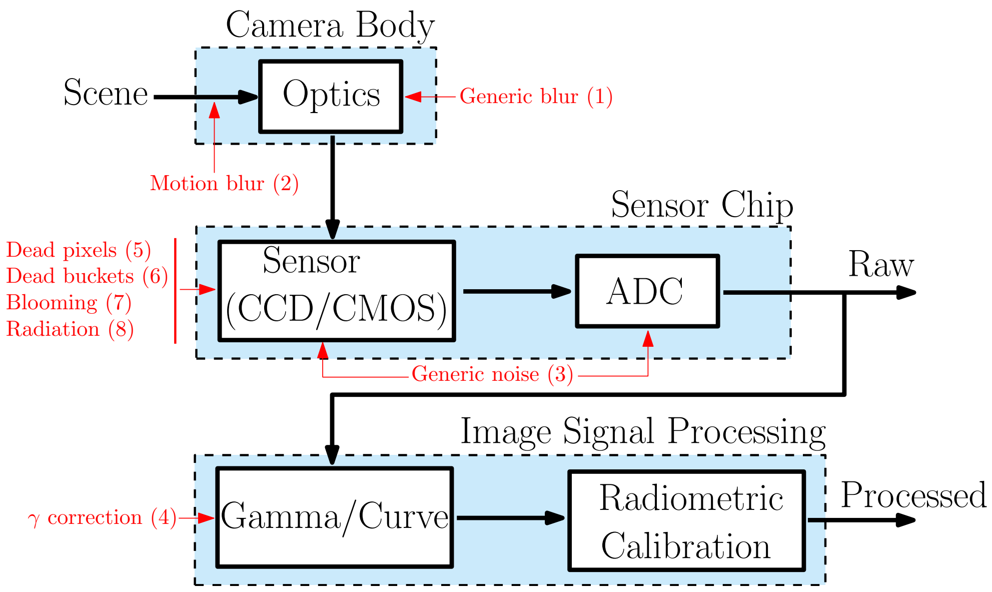


Figure 3.12: Noise modeling block in CORTO. Adapted from [22]. Different noise sources are represented in red with the order in which they are applied in the script.

The block is currently implemented in Matlab and the different sources of noise that the user can parametrize are represented in red in Figure 3.12, spread into 8 different steps. In (1), generic blur is introduced with the use of a Gaussian filter from the *imgaussfilt* function, while motion blur is simulated in (2), generating a specific motion filter with the *fspecial* function. In (3), generic noise is added with the *imnoise* function. In (4), gamma correction is performed over the image using the *imadjust* function. After these phenomena are modeled, sensor effects are introduced in (5), (6), and (7) by altering the pixel values in the images, removing single pixels, entire rows and columns, or saturating pixel content. Finally, in (8), radiation effects are introduced simulating randomized lines spanning over the image saturating pixel's content.

The different sources of noise illustrated in Figure 3.12 are exemplified on an image of Dimorphos at low (top), medium (center), and high (bottom) values in the mosaic image in Figure 3.13 (except for dead pixels, dead buckets, and blooming effects, which are illustrated from top to bottom in the second column from the right).

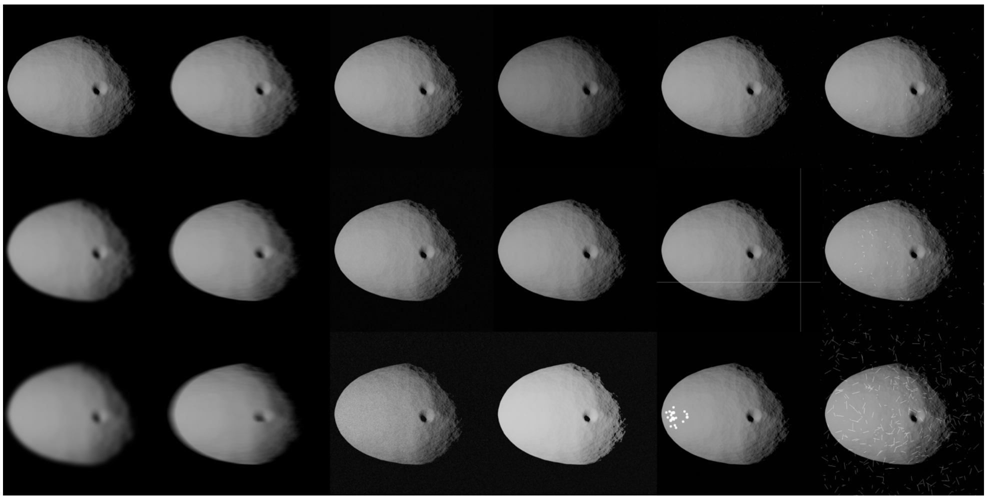
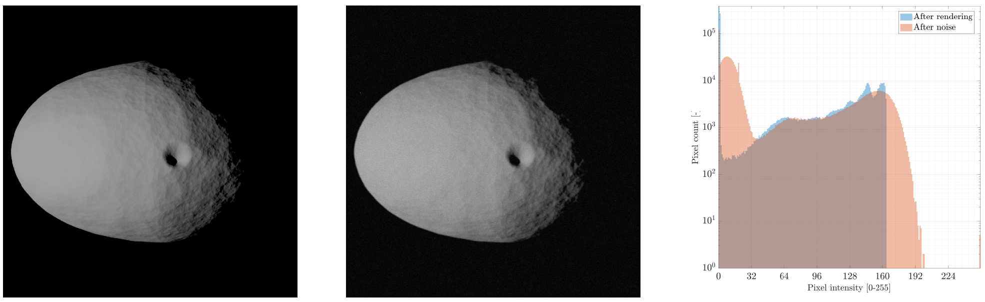


Figure 3.13: Example of different sources of noise, low-to-high from top to bottom. From left to right: Generic blur, motion blur, generic Gaussian noise, Gamma correction, sensor effects (from top to bottom: dead pixels, dead buckets, and blooming), and radiation effects.

The effects of noise modeling are visualized in histogram form in Figure 3.14. These effects are important to change the image content of a clean synthetic sample toward a more realistic one.



(a) Synthetic image after rendering, without noise. **(b)** Synthetic image after application of noise. **(c)** Histogram comparison with and without noise.

Figure 3.14: Effects of the application of noise to the synthetic images. The binwdith of the histogram is equal to 1.

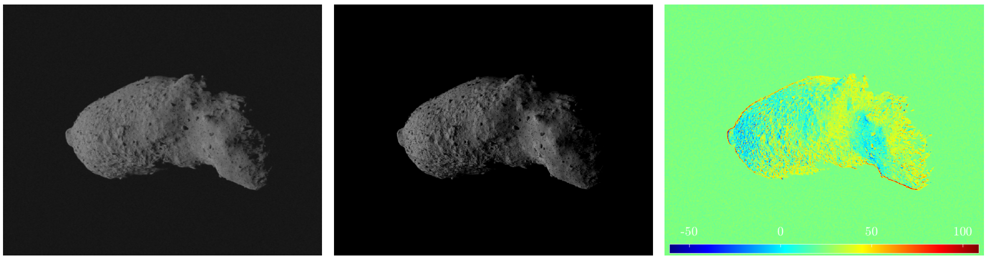
Finally, it is mentioned that the noise modeling block can be used in two pragmatic approaches. First, one could mimic the target noise effects expected from a specific camera for a pre-designed mission. Second, one could also apply noise with statistical sampling from a pre-defined distribution to generate datasets with images with generic and realistic noise that is not typical of any sensor. The latter methodology tends to perform better with a data-driven approach in terms of domain randomization, increasing their robustness and applicability to real images.

3.1.1.4 Hardware-in-the-loop

CORTO can also be used to provide synthetic images to the TinyV3RSE, an optical testbench facility. TinyV3RSE [98, 99] comprises a high-resolution screen, a plano-convex collimator lens, and a camera. More details about the facility are illustrated in Section 3.2.

The facility's purpose is twofold. First, including an engineering model of the target camera can be useful in identifying unforeseen bottlenecks and preparing the correct interface. Second, the camera transforms an otherwise clean Blender image into a noisy one, with characteristics similar to a real deployment.

A comparison between a synthetic image and its equivalent image captured within the TinyV3RSE facility is illustrated in Figure 3.15. The two images are photometrically different (the camera-screen interaction introduces noise, and photometric calibration is not performed in TinyV3RSE) but geometrically equivalent, as it is possible to see from the image difference in Figure 3.15.



(a) Synthetic image. **(b)** Facility image. **(c)** Synthetic-Facility.

Figure 3.15: Geometric equivalence between synthetic and facility images.

3.1.1.5 Post-processing

A postprocessing block in CORTO adapts the image-label pairs for the specific target application. This is particularly relevant when developing data-driven image processing methods, such as convolutional neural networks, that traditionally require constant size tensors for training, validation, and testing.

The image obtained at rendering often cannot be used directly as input of the IP methods, especially data-driven ones, for three main reasons. First, the original image resolution is too high. Due to hardware limitations, image size needs to be reduced. This is typical of IP methods based on convolutional networks, which could encounter memory or processing saturation issues if working with images at native resolutions. Second, the ideal pointing often assumed during rendering simplifies image generation but causes poor variability of the input-label relationship, which can cause poor generalization capability of the IP methods in case of real pointing conditions. Third, images of celestial bodies are often populated by empty backgrounds, especially when seen from relatively far away. In these cases, cropping has been demonstrated to improve performances of the IP methods over the resizing of the images.

All three issues are addressed together in a unique postprocessing pipeline that transforms an image and its associated labels from a geometrical and rendering space referred to as $\mathbb{S}_0$ into a new space $\mathbb{S}_2$. A sketch of the pipeline is illustrated in Figure 3.16. Currently, only labeling changes for the center of mass, center of brightness, and range from the body are supported, but other labels can be easily included in the pipeline if needed. Also, the pipeline processes squared images, but its extension to rectangular ones is trivial.

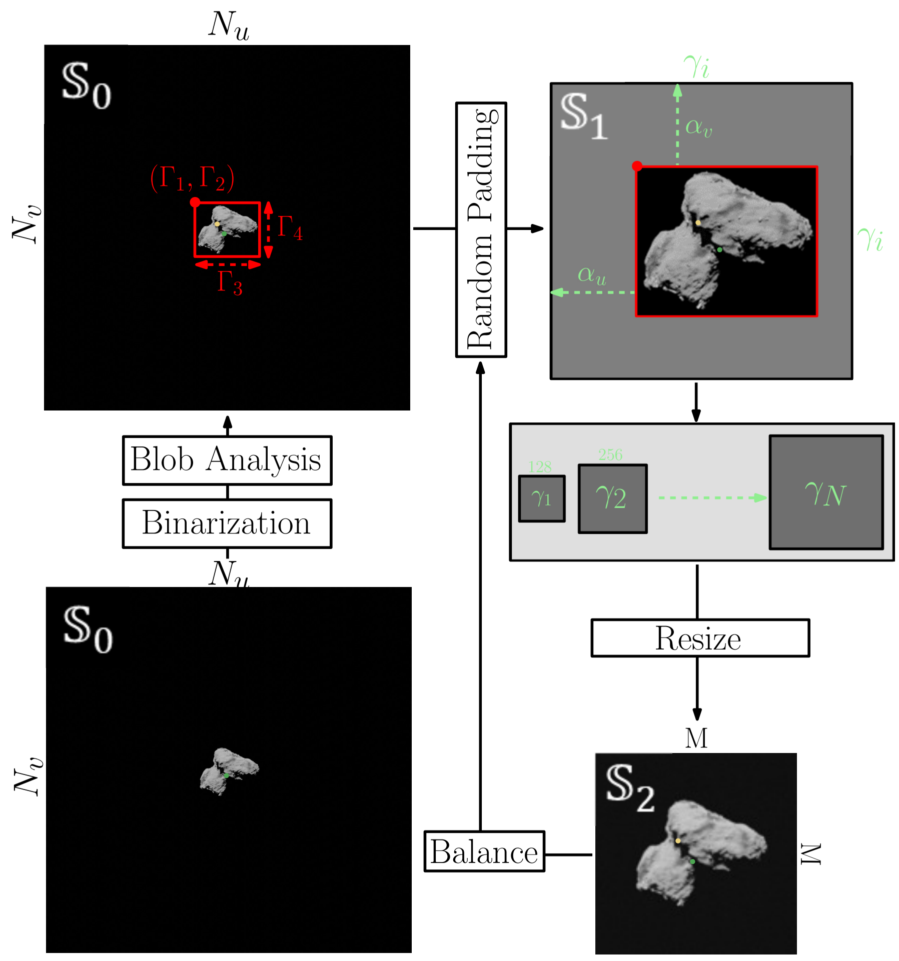


Figure 3.16: High-level architecture of the postprocessing pipeline.

The original image obtained after rendering and noise addition is said to be defined in a $\mathbb{S}_0$ space with a resolution $N_u \times N_v$. The image is binarized using [106], and a simple blob analysis is performed. The N-th biggest blobs are identified and grouped to form a single object with a bounding box Γ (defined by the top-left corner coordinates Γ_1 and Γ_2 and by the width Γ_3 and height Γ_4). Random padding of the image outside of the bounding box Γ is performed to transform the $\Gamma_1 \times \Gamma_2$ bounding box into a square of resolution $\gamma_i \times \gamma_i$ in $\mathbb{S}_1$ space. The padding is performed by randomly sampling two scalars α_u and α_v from uniform distributions, each ranging from *0* to the maximum value that would transform the rectangular $\Gamma_1 \times \Gamma_2$ box into a square one of side γ_i. During padding, no new pixel content is generated, and the same pixel content from the original image in $\mathbb{S}_0$ is instead retrieved. Also, note that the target size γ_i of the image in $\mathbb{S}_0$ can be selected by the user as a multiple of the final target size M. Finally, the image is resized and transformed into a $M \times M$ matrix, defined in $\mathbb{S}_2$. Lastly, a final step can be performed to repeat the random padding process multiple times. This may be necessary if the user is interested in balancing the dataset's image appearance or label distributions and would like to implement different input distributions.

All these steps are part of a data augmentation strategy designed explicitly for celestial bodies and their interest labels. Traditional data augmentation strategies turned out to be limiting the capabilities to transform the image-label pairs in a useful way. It is also mentioned that only the procedure for changing the image is illustrated in Figure 3.16, but the labels are changed simultaneously to ensure they can be correctly recovered. For example, performing the $\mathbb{S}_0 \rightarrow \mathbb{S}_1$ transformation from $\mathbb{S}_0$ to $\mathbb{S}_1$ results in a change in the **CoB** and **CoF** coordinates:

$$\mathbf{CoB}^{\mathbb{S}_1} = \mathbf{CoB}^{\mathbb{S}_0} - \begin{bmatrix} \Gamma_1 \\ \Gamma_2 \end{bmatrix} + \begin{bmatrix} \alpha_u \\ \alpha_v \end{bmatrix} \tag{3.1}$$

$$\mathbf{CoF}^{\mathbb{S}_1} = \mathbf{CoF}^{\mathbb{S}_0} - \begin{bmatrix} \Gamma_1 \\ \Gamma_2 \end{bmatrix} + \begin{bmatrix} \alpha_u \\ \alpha_v \end{bmatrix} \tag{3.2}$$

while the other labels, such as the **CoF-CoM** offset, the range from the target body, the position in cartesian coordinates, or polar coordinates remain unchanged:

$$\delta^{\mathbb{S}_1} = \delta^{\mathbb{S}_0} \tag{3.3}$$

$$\rho^{\mathbb{S}_1} = \rho^{\mathbb{S}_0} \tag{3.4}$$

$$[X, Y, Z]^{\mathbb{S}_1} = [X, Y, Z]^{\mathbb{S}_0} \tag{3.5}$$

$$[\phi_1, \phi_2]^{\mathbb{S}_1} = [\phi_1, \phi_2]^{\mathbb{S}_0} \tag{3.6}$$

Performing the $\mathbb{S}_1 \rightarrow \mathbb{S}_2$ transformation from $\mathbb{S}_1$ to $\mathbb{S}_2$ the labels are transformed as follows:

$$\mathbf{CoB}^{\mathbb{S}_2} = \mathbf{CoB}^{\mathbb{S}_1} \frac{128}{\gamma} \tag{3.7}$$

$$\mathbf{CoF}^{\mathbb{S}_2} = \mathbf{CoF}^{\mathbb{S}_1} \frac{128}{\gamma} \tag{3.8}$$

$$\delta^{\mathbb{S}_2} = \delta^{\mathbb{S}_1} \frac{128}{\gamma} \tag{3.9}$$

$$\rho^{\mathbb{S}_2} = \rho^{\mathbb{S}_1} \frac{128}{\gamma} \tag{3.10}$$

$$[X, Y, Z]^{\mathbb{S}_2} = [X, Y, Z]^{\mathbb{S}_1} \frac{128}{\gamma} \tag{3.11}$$

while ϕ_1 and ϕ_2 remain unchanged: $[\phi_1, \phi_2]^{\mathbb{S}_2} = [\phi_1, \phi_2]^{\mathbb{S}_1}$. Note that the phase angle is a quantity that remains unchanged during the transformations.

Concatenating the transformations $\mathbb{S}_0 \rightarrow \mathbb{S}_1$ and $\mathbb{S}_1 \rightarrow \mathbb{S}_2$ and viceversa is straightforward. $\mathbb{S}_0 \rightarrow \mathbb{S}_2$ can then be performed on a batch of images if these are being grouped in a dataset or on a single image if the steps are being processed onboard while transforming from an acquisition at camera resolution to an image at a processing resolution. The image-label pairs can be used in $\mathbb{S}_2$ for training purposes, while the image alone can be used in inference. In the latter case, the labels estimated in $\mathbb{S}_2$ can be transformed back to meaningful labels in $\mathbb{S}_0$ using the inverse transformation $\mathbb{S}_2 \rightarrow \mathbb{S}_0$.

This is possible because the pipeline has been implemented specifically for onboard implementation, ensuring non-destructive operations are performed over the image-label pairs and since all necessary quantities (γ, Γ_1, Γ_2, α_u, α_v, $CoB_u^{\mathbb{S}_0}$, and $CoB_v^{\mathbb{S}_0}$) can be easily stored.

Finally, as image-label pairs are often used to estimate a position with respect to the target body, the necessary transformations to transform the labels illustrated above into positions are briefly described. When an image is used to extract the cartesian coordinates $[X, Y, Z]$ in $\mathbb{S}_2$ in any reference frame, these are transformed back to $\mathbb{S}_0$ as:

$$\mathbf{p}_{est}^{\mathbb{S}_0} = \frac{\gamma}{128} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{est}^{\mathbb{S}_2} \tag{3.12}$$

On the other hand, the same transformation using polar coordinates is:

$$\mathbf{p}_{est}^{\mathbb{S}_0} = \Omega \left(\begin{bmatrix} \phi_1 \\ \phi_2 \\ \rho \frac{\gamma}{128} \end{bmatrix}_{est}^{\mathbb{S}_2} \right) \tag{3.13}$$

where Ω represents the trivial transformation function from spherical to cartesian coordinates. Finally, any IP method working in inference with δ, ρ labels requires a set of intermediate steps to generate a position estimate. The following optical observations are generated in inference in $\mathbb{S}_0$:

$$\mathbf{o}_{est}^{uv,\mathbb{S}_0} = \begin{bmatrix} CoF_{est,u}^{\mathbb{S}_0} \\ CoF_{est,v}^{\mathbb{S}_0} \\ 1 \end{bmatrix} = \begin{bmatrix} CoB_u^{\mathbb{S}_0} + \delta_u{}^{\mathbb{S}_2}\frac{\gamma}{128} \\ CoB_v^{\mathbb{S}_0} + \delta_v{}^{\mathbb{S}_2}\frac{\gamma}{128} \\ 1 \end{bmatrix} \tag{3.14}$$

$$\rho_{est}^{\mathbb{S}_0} = \rho^{\mathbb{S}_2}\frac{\gamma}{128} \tag{3.15}$$

where $\delta_u^{\mathbb{S}_2}$, $\delta_v^{\mathbb{S}_2}$, and $\rho^{\mathbb{S}_2}$ are the output of the IP method while all other variables are computed and stored during the processing of the image by the pipeline. The vector of optical observables $\mathbf{o}_{est}^{uv,\mathbb{S}_0}$ is then transformed from the $\mathcal{UV}$ reference frame, which expresses pixel coordinates on the image to the Image Plane (ImP) reference frame using the inverse of the camera calibration matrix $\mathbf{K}^{-1}$ defined with the notation in [107]:

$$\mathbf{o}_{est}^{ImP} = \mathbf{K}^{-1}\mathbf{o}_{est}^{uv,\mathbb{S}_0} \tag{3.16}$$

The $\mathbf{o}_{est}^{ImP}$ vector is then transformed into a LoS vector in the $\mathcal{CAM}$ reference frame. Using the range $\rho^{\mathbb{S}_2}$, this LoS can be transformed into a position estimate in the camera reference frame $\mathbf{p}_{est}^{\mathcal{CAM}}$. If the position needs to be expressed in a different reference frame than the camera one, a change of reference frame is required. This can be performed using the attitude quaternion of the spacecraft, assuming it to be known relatively accurately from the attitude determination process of a Star-tracker and to know the rigid transformation between additional intermediate frames (such as the ones between spacecraft body reference frame and camera mounting). Further reference frame transformation can be stacked together if the transformations between them are known or can be retrieved. The position estimate into a generic reference frame $\mathcal{RF}$ can thus be written as:

$$\mathbf{p}_{est}^{\mathcal{RF},\mathbb{S}_0} = \mathbf{q}_{\mathcal{CAM}\rightarrow\mathcal{RF}}\mathbf{p}_{est}^{\mathcal{CAM}} \tag{3.17}$$

where the quaternion $\mathbf{q}_{\mathcal{CAM}\rightarrow\mathcal{RF}}$ stacks together all the attitude transformations needed to pass from the $\mathcal{CAM}$ to the $\mathcal{RF}$ one.

Finally, thanks to the postprocessing strategy just described, generating multiple versions from the same input image with different image-label pairs is possible. This can be useful for performing data augmentation for generalization purposes and implementing domain randomization designs.

3.1.1.6 Reproduce previously flown missions

Recreating images obtained from previously flown missions holds tremendous potential since it enables direct comparison with previously flown algorithms using the same type of input images.

For this reason, a tool has been designed to recreate images taken from previous sensors. The tool uses the available metadata associated with the existing images to extract the epoch and, combining this with the camera properties and the spacecraft-body-Sun relative poses, generate a set of inputs that can be used in

CORTO. The objects poses are retrieved from kernels using SPICE [20]. At the current stage of development, this tool can replicate images taken from missions led by major space agencies, such as ESA, National Aeronautics and Space Administration (NASA), and Japan Aerospace Exploration Agency (JAXA).

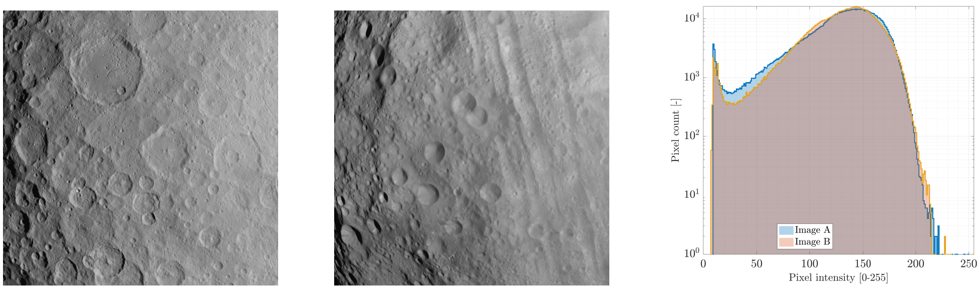
Note that images reproduced with this tool are accompanied by the same attitude estimation and positioning errors reflected in the ephemerides. This, in turn, translates into minor errors in the labels, which need to be considered.

3.1.2 Validation

Validating synthetic images of celestial objects with real ones is crucial in ensuring synthetic images' accuracy, reliability, and applicability for various scientific and operational purposes.

While image histogram comparison is often used as the primary method to assess the similarity between synthetic and real images [108–110], it represents the satisfaction of a necessary but not sufficient condition. An image histogram represents the distribution of the image content across different intensities, but in doing so, it inevitably brings a loss of spatial information.

For example, in Figure 3.17(c), two images of two different asteroids, namely Ceres and Vesta, are illustrated with their image histograms overlapped. These images have been captured by the Dawn [14] mission using the same camera. Albeit their histograms are similar and exhibit a consistent overlap, it cannot be concluded that the two images correctly represent the same scenery, showcasing how risky it is to adopt this criterion alone when assessing image similarity for validation purposes. The histogram overlap describes a similarity in the pixel content of the image, which does not reflect a similar spatial distribution of such content.



(a) Image A. **(b)** Image B. **(c)** Histograms.

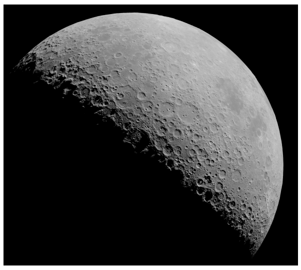
Figure 3.17: Image of Ceres (a) and Vesta (b) along with their overlapping histograms (c). The binwdith of the histogram is equal to 1.

Albeit it is important to check that similar images exhibit similar histograms, an approach should be used instead that can better quantify image differences.

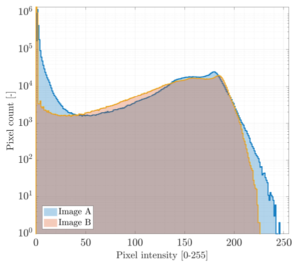
[20]https://naif.jpl.nasa.gov/naif/data.html, last accessed: 8th of August, 2023.

A manual approach would adjust the rendering settings and the relative body-camera-Sun poses. For instance, Figure 3.18 shows a manually reproduced image of the Moon seen with a full limb compared to one captured on the Orion spacecraft with a navigation camera [21]. Albeit this approach can yield faithful reproduction of real images, it demands a significant amount of time and introduces human errors. To overcome these limitations, a systematic approach is proposed instead in this section to validate the functionalities of CORTO.

(a) Image A.



(b) Image B.



(c) Histograms.

Figure 3.18: Real (a) and synthetic (b) manually reproduced images of the Moon along with their overlapping histograms (c). The binwdith of the histogram is equal to 1.

3.1.2.1 Pipeline

This section proposes a validation pipeline as a systematic approach to evaluate image similarity considering pixel intensity values and overall image structure.

A schematic of the validation pipeline is represented in Figure 3.19. The pipeline inputs are the real image and N template images generated using varying settings in CORTO. These may include rendering, shading, material, surface, and light properties.

As depicted in Figure 3.19, the first operation involves a normalized cross-correlation [111] between the N templates and the real image to reduce the camera poses errors introduced by the state reconstruction, computed as:

$$\gamma(u,v) = \frac{\sum_{x,y}[f(x,y)-\bar{f}_{u,v}][t(x-u,y-v)-\bar{t}]}{\{\sum_{x,y}[f(x,y)-\bar{f}_{u,v}]^2\sum_{x,y}[t(x-u,y-v)-\bar{t}]^2\}^{0.5}} \tag{3.18}$$

where (x,y) denotes the pixel location, f is the real image, $\bar{t}$ is the mean of the template, (u,v) are the coordinates of the template center in the real image, and $\bar{f}_{u,v}$ is the mean of $f(x,y)$ within the template region. Following this operation, the images are cropped to maximize the correlation, which is particularly significant for far-range observations, where the complete silhouette of the body is visible. The outcome of this process yields N cropped template images along with their

[21]https://www.nasa.gov/image-feature/orion-gazes-at-moon-before-return-to-earth, last accessed 8th of August, 2023.

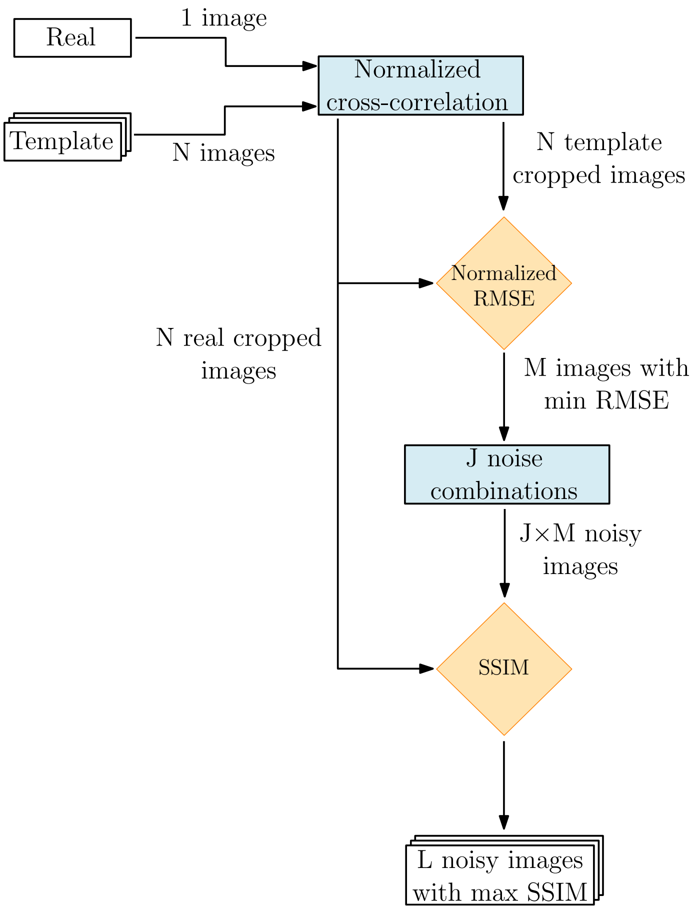


Figure 3.19: High-level architecture of the validation pipeline.

corresponding cropped real images, such that each template has a corresponding real image with the same resolution. Subsequently, the cropped real and template images are compared with a Normalized Root Mean Square Error (NRMSE) [112], computed as:

$$\mathrm{NRMSE} = \frac{\sqrt{\frac{1}{d}\sum_{x',y'}[t(x',y') - f(x',y')]^2}}{d} \tag{3.19}$$

where d represents the number of pixels and (x',y') denotes the pixel location in the cropped images. The normalization approach has been selected because the previous correlation step generates different-sized images. Consequently, the value of the RMSE is scaled to be independent of the image size for a better comparison.

The first M images (with the lowest NRMSE values) are selected because of the similarities in pixel intensities. Once the best ideal synthetic images are selected, accounting for noise inherent to the environment and camera errors is necessary. Because of this, J different noise combinations are applied to each of the M images.

Specifically, the considered noise source includes Gaussian noise mean and variance, blur, and brightness. The assumed noise values are specified in Table 3.1, resulting in *192* combinations.

Table 3.1: Considered noise values to be applied to the M template images.

Noise type	Values
Gaussian mean	$0.01, 0.09, 0.17, 0.25$
Gaussian variance	$10^{-5}, 10^{-4}, 10^{-3}$
Blur	$0.6, 0.8, 1.0, 1.2$
Brightness	$1.00, 1.17, 1.33, 1.50$

As a result, a total of $J \times M$ noisy images become available. Lastly, a second comparison step uses the Structural Similarity Index (SSIM) [113]. This metric is employed because it considers the structural information embedded in the image, separating it from the influence of the illumination. The metric is defined as:

$$\mathrm{SSIM}(x', y') = [l(x', y')]^{\alpha}[c(x', y')]^{\beta}[s(x', y')]^{\gamma} \tag{3.20}$$

where l, c, and s represent the image's luminance, contrast, and structural terms, respectively. The coefficients α, β, and γ are all set to *1* to ensure equal contribution. After evaluation by the SSIM, the L images with maximum SSIM are selected as the best validation candidate.

3.1.2.2 Results

Using the validation pipeline described in the previous section, the tool's capability is validated considering four minor bodies: Ceres, Vesta, Bennu, and 67P. These target bodies have been selected because their images are readily available and since they represent a diverse sample in terms of global shape and surface characteristics.

Different strategies to represent the surface are investigated for each body, referred to as: "OSL" if specific scattering laws (as the one designed in [92] and presented in Section 3.1.1.2) are used, "PBSDF" if the standard Blender scattering function is used in the shader (without a texture), and "PBSDF + Texture" if an existing body texture has been used instead, coupled with the PBSDF.

Table 3.2 summarizes the total number of template images for each combination considered. Note that some cases are not considered in the pipeline (e.g., Vesta and 67P with Texture, or Ceres with only OSL or PBSDF) as these settings would not be considered for representing these bodies, as the texture maps would not be available or the scattering functions alone would not be capable of yielding the desired fidelity. The template images from Table 3.2 are obtained by varying the illumination intensity of the Sun's lamp while changing the albedo's properties of the target body and the scattering function adopted. Finally, it is remarked that the case of asteroid Bennu has been investigated across all possible reflectance models since the existence of a high-quality and low phase-angle texture map of Bennu allows for such a detailed comparison.

Table 3.2: Number of template images for each combination of target body and scattering function used to represent the surface.

	OSL	PBSDF	PBSDF + Texture
Ceres	-	-	693
Vesta	8316	-	-
67P	1512	-	-
Bennu	2646	686	819

The validation results of the pipeline over these bodies are presented both in a quantitative way in Table 3.3 and Table 3.4 using the SSIM similarity metric, and in a visual way in Figure 3.20 and Figure 3.21. The link between the tables and the mosaic views is represented by the row and column coordinates listed in the last column of each table. Moreover, Table 3.3 and Table 3.4 provide details about each sample, including the original name of the image, the key rendering properties used (the scattering function, from *0* associated to the PBSDF, to *6* following the order illustrated in Section 3.1.1.2, albedo, and Sun's intensity), the noise combination (expressed as four components respectively for Gaussian mean and variance, blur, and Brigthness as in Table 3.1), and the associated SSIM with the most similar synthetic image.

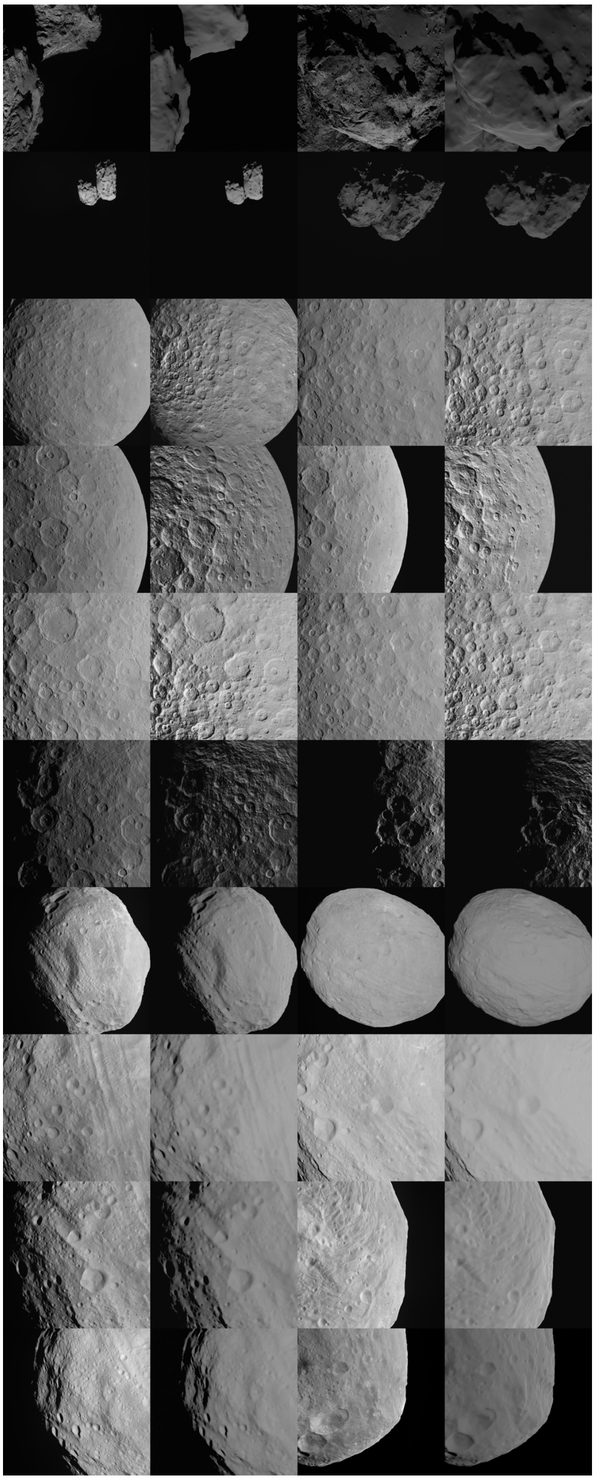


Figure 3.20: Mosaic view of synthetic and real images of 67P, Ceres, and Vesta. The first and third columns represent real images, while the second and fourth ones are generated using CORTO.

Table 3.3: Properties of the synthetic images of comet 67P and asteroids Ceres and Vesta. The first four rows correspond to 67P, five to twelve represent Ceres, and thirteen to twenty refer to Vesta. The last column represents the position as (row, column) coordinates of the synthetic image in Figure 3.20.

Img Name	Rendering	Noise	SSIM	ID
N20160128T002344268ID20F71	$4,0.15,40$	$0.01,10^{-5},1.2,1.00$	0.7537	1,1
N20160130T173323717ID20F22	$6,0.15,30$	$0.01,10^{-5},1.2,1.00$	0.4421	1,3
W20150316T053347931ID20F13	$3,0.5,40$	$0.09,10^{-5},1.2,1.50$	0.9360	2,1
W20160617T102200832ID20F18	$5,0.5,10$	$0.09,10^{-5},1.2,1.33$	0.8920	2,3
FC21A0037273_15136172940F1E	$0,-,4.25$	$0.09,10^{-3},0.8,1.00$	0.4430	3,1
FC21A0037405_15157034032F3I	$0,-,4.20$	$0.09,10^{-3},0.6,1.50$	0.3979	3,3
FC21A0037589_15158013232F1I	$0,-,3.25$	$0.09,10^{-3},0.8,1.00$	0.3558	4,1
FC21A0037593_15158020232F1I	$0,-,6.50$	$0.09,10^{-3},0.8,1.00$	0.4973	4,3
FC21A0037978_15163064254F1G	$0,-,6.25$	$0.09,10^{-3},0.6,1.50$	0.2870	5,1
FC21A0038693_15172150728F6G	$0,-,6.00$	$0.09,10^{-3},0.6,1.50$	0.3557	5,3
FC21A0038787_15173122643F1G	$0,-,1.50$	$0.01,10^{-5},1.2,1.00$	0.3379	6,1
FC21A0039042_15176210244F1H	$0,-,2.50$	$0.09,10^{-3},0.8,1.17$	0.6744	6,3
FC21B0003258_11205095604F6C	$6,0.5,3$	$0.09,10^{-3},0.8,1.17$	0.7201	7,1
FC21B0003428_11205235222F5C	$6,0.5,2$	$0.09,10^{-3},0.8,1.17$	0.8499	7,3
FC21B0003757_11218102757F7D	$3,0.5,2$	$0.09,10^{-3},0.8,1.00$	0.7034	8,1
FC21B0003866_11218121551F4D	$5,0.5,3$	$0.09,10^{-3},0.6,1.50$	0.8337	8,3
FC21B0004630_11226232738F7D	$6,0.5,2$	$0.09,10^{-3},0.6,1.33$	0.6480	9,1
FC21B0005299_11230130409F6B	$6,0.5,1$	$0.09,10^{-3},0.8,1.17$	0.8114	9,3
FC21B0005871_11232204234F4B	$2,0.5,2$	$0.01,10^{-3},1.2,1.50$	0.6644	10,1
FC21B0006422_11238100914F1B	$6,0.5,1$	$0.01,10^{-4},1.0,1.33$	0.8142	10,3

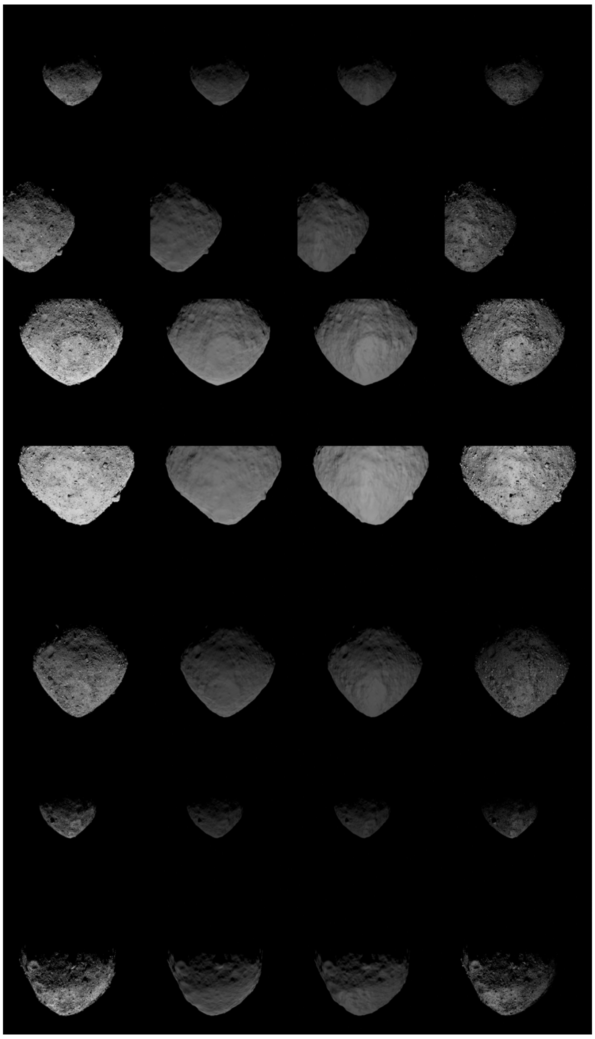


Figure 3.21: Mosaic view of Bennu. The first column represents real images, while progressing from the second to the last column, synthetically generated images are depicted using CORTO with the OSL reflectance models, PBSDF, and PBSDF + Texture, respectively.

Table 3.4: Properties of the synthetic images of Bennu. For every real image presented in the first column, three rows provide information about the corresponding synthetic image properties, namely, OSL, PBSDF, and PBSDF + Texture. The last column represents the coordinate as (row, column) of the synthetic image in Figure 3.21.

Img Name	Rendering	Noise	SSIM	ID
	$6, 0.15, 40$	$0.01, 10^{-4}, 1.0, 1.33$	0.9200	1,2
20181211T181336S699_map_specradL2b	$0, -, 0.40$	$0.01, 10^{-4}, 1.0, 1.33$	0.9187	1,3
	$0, -, 3.75$	$0.01, 10^{-4}, 0.8, 1.00$	0.9458	1,4
	$6, 0.50, 20$	$0.01, 10^{-4}, 1.0, 1.33$	0.8054	2,2
20181212T043459S572_map_specradL2x	$0, -, 0.60$	$0.01, 10^{-4}, 1.0, 1.33$	0.8046	2,3
	$0, -, 6.00$	$0.01, 10^{-4}, 0.8, 1.00$	0.8958	2,4
	$6, 0.50, 40$	$0.01, 10^{-4}, 1.0, 1.33$	0.7090	3,2
20181212T064255S344_map_radL2pan	$0, -, 1.20$	$0.01, 10^{-4}, 1.0, 1.33$	0.7078	3,3
	$0, -, 14.75$	$0.01, 10^{-4}, 0.8, 1.00$	0.8549	3,4
	$6, 0.50, 40$	$0.01, 10^{-4}, 1.0, 1.33$	0.7131	4,2
20181212T085936S404_map_iofL2pan	$0, -, 2.00$	$0.01, 10^{-4}, 1.0, 1.33$	0.7165	4,3
	$0, -, 23.5$	$0.01, 10^{-4}, 0.8, 1.00$	0.8459	4,4
	$6, 0.20, 40$	$0.01, 10^{-4}, 1.0, 1.33$	0.7537	5,2
20181213T043620S487_map_radL2pan	$0, -, 0.60$	$0.01, 10^{-4}, 0.8, 1.50$	0.7528	5,3
	$0, -, 5.00$	$0.01, 10^{-4}, 0.8, 1.00$	0.8175	5,4
	$3, 0.15, 35$	$0.01, 10^{-4}, 1.0, 1.33$	0.9339	6,2
20181215T053926S725_map_iofL2b	$0, -, 0.30$	$0.01, 10^{-4}, 1.0, 1.33$	0.9337	6,3
	$0, -, 3.00$	$0.01, 10^{-4}, 0.8, 1.00$	0.9473	6,4
	$6, 0.45, 20$	$0.01, 10^{-4}, 1.0, 1.33$	0.8127	7,2
20181217T033612S897_map_iofL2pan	$0, -, 0.60$	$0.01, 10^{-4}, 1.0, 1.33$	0.8107	7,3
	$0, -, 6.50$	$0.01, 10^{-4}, 0.8, 1.00$	0.8845	7,4

From the results of the validation pipeline, from a qualitative inspection, it is possible to determine that CORTO can represent the target bodies as realistic input for image processing and visual-based applications. Remarkably, the use of texture maps yields mixed results for the cases of Ceres and Bennu. In particular, considering all the reflectance strategies for Bennu, the one using "PBSDF + Texture" turns out to be the one returning the highest similarity scores, as can be seen by the higher values of SSIM in Table 3.4. On the other hand, the values of SSIM for the case of Ceres are lower than expected, even when using a texture map over the surface. This difference is ascribed to the quality of the available textures and the data they represent (e.g., mostly craters and albedo variations for Ceres and boulders and albedo variations for Bennu). This indicates that whenever this information is available at high resolution, it can significantly improve the similarity of synthetic images.

Moreover, the effect is more relevant when representing boulder fields over plain and cratered regions. Unfortunately, this form of data is solely available at high resolution and with the correct illumination conditions only for a limited number of bodies. Nonetheless, when texture maps are not used, as in the case of Vesta and 67P, CORTO can represent the appearance of the bodies at global level with a reasonable level of fidelity.

3.1.3 Case studies

Due to its capabilities, CORTO has been concurrently developed by the author and used within the DART group since Summer 2020 in various research activities, projects, and missions.

CORTO has been extensively used and co-developed for the design and validation of the image processing and visual-based GNC of the Milani CubeSat (see Section 6), a 6U CubeSat that will visit the Didymos binary system in 2027 as part of the ESA Hera mission [114]. The tool proved to be critical in the design of the data-driven algorithms within the image processing of Milani, in the testing of the object recognition algorithm, and in validating all the visual-based applications of the GNC subsystem of the CubeSat.

CORTO capabilities have also been used to test the limb-based navigation around the Moon for the LUnar Meteoroid Impact Observer (LUMIO) mission [115]. LUMIO is a 12U CubeSat that will orbit in a Halo orbit about the Earth-Moon L2 point and is an ASI/ESA mission. LUMIO will perform a scientific investigation about meteoroid impacts and act as a technology demonstrator for visual-based navigation techniques.

Additionally, up-to-date CORTO is currently used in different ASI and ESA projects, most notably the DeepNav [116] and StarNav projects. DeepNav is currently investigating the design and implementation of deep-learning techniques for visual-based navigation around small bodies using onboard processors. All the datasets used within the DeepNav project for the training and testing of the deep-learning methods have been generated with CORTO and using a HIL setup with the TinyV3RSE facility. Finally, StarNav is an ongoing project investigating star trackers' image-processing capabilities for close proximity operations about asteroids and the Moon. The test images for the image-processing algorithms have been generated using CORTO.

3.2 Hardware-in-the-loop with an optical facility

TinyV3RSE is an optical vision-based algorithm test-bench that has been designed within the DART lab to promote fundamental research on spacecraft autonomy and in support of validation of the IP and visual-based navigation algorithms in which the DART lab is responsible for the design.

TinyV3RSE being a compact, low-cost, versatile HIL setup, it allows simple camera-in-the-loop testings with noisy images.

3.2.1 Design

TinyV3RSE design results from a continuous and collective effort since the beginning of 2019. An original facility design has been performed as part of the work illustrated in [117]. Hardware and software changes have been implemented over the years, improving the original design, as described in [99, 118, 119]. Finally, software changes in refined calibration procedures have been perfectioned in [98]. At the same time as these development activities, the facility proved fundamental as a validation tool in different projects, missions, and research activities. What follows is a brief description of the critical design elements of TinyV3RSE, followed by a detailed description of its core components.

TinyV3RSE comprises three main elements: a screen, a collimator, and a camera, positioned as in Figure 3.25. When the light emitted by the screen passes through the collimating lens, it respects the thin lens equation:

$$\frac{1}{f_{\text{coll}}} = \frac{1}{d_{\text{r}}} + \frac{1}{d_{\text{i}}} \tag{3.21}$$

where f_{coll} is the collimating lens focal length, d_{i} is the distance between the collimating lens and the image, and d_{r} is the distance between the collimating lens and the object, i.e., the screen. Recall that f_{coll} is positive for converging lenses and negative for diverging ones. Moreover, note that d_{r} is positive when placed on the left side of the lens and negative otherwise. Finally, d_{i} is positive when the image is generated on the right side of the lens, i.e., a real image is formed, and negative otherwise, i.e., a virtual image is formed. Figure 3.22 shows the geometrical configuration under study, which generates a virtual image in this case.

Equation 3.21 can be rewritten to compute the image distance explicitly as:

$$d_{\text{i}} = \left(\frac{d_{\text{r}}}{d_{\text{r}} - f_{\text{coll}}}\right) f_{\text{coll}} \tag{3.22}$$

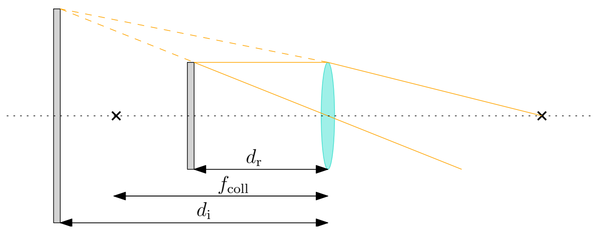


Figure 3.22: Geometrical configuration for the lens equation of the collimator in the case considered in TinyV3RSE.

Equation 3.22 shows that the observed object (i.e., the screen) must be placed at the focal length distance to be seen as from infinity. The perfect design choice would be for the screen's image to fit the camera FOV fully. However, to simplify the design process, the vertical FOV is considered instead since it is smaller than the horizontal one for the target cameras to use within the facility. Under the assumption of perfect components' alignment, the problem can be framed as outlined in Figure 3.23. Thanks to fundamental geometrical relationships, it is easy to demonstrate that:

$$\tan\left(\frac{\theta}{2}\right) = \frac{h_s}{2f_{coll}} \tag{3.23}$$

where θ is the camera FOV, h_s is the vertical screen size, and $\theta_1 = \theta_2 = \frac{\theta}{2}$. Equation 3.23 links the three components of TinyV3RSE, showing that their design choice is not arbitrary.

Note that, because of the collimation, the distance between the camera and the collimating lens d_{cam} is not a design parameter that depends on the collimator focal length. This parameter is essential to determine the diameter of the collimating lens [120]. To avoid the camera observing outside of the collimating lens, the following relation has to be satisfied:

$$R_{coll} \leq R_{cam} + d_{cam} \tan\left(\frac{\theta}{2}\right) \tag{3.24}$$

where R_{coll} is the collimating lens radius and R_{cam} is the camera lens objective radius. Moreover, to ensure it works in the paraxial area of the collimating lens, i.e., where the thin lens equation hypothesis holds, d_{cam} must be chosen as small as possible.

3.2.2 Components

TinyV3RSE is composed of three main modules mounted on an enclosed optical table:

1. **The camera**, rigidly mounted on its mechanical support, which enables vertical translation, pitch, and yaw mechanical adjustments;

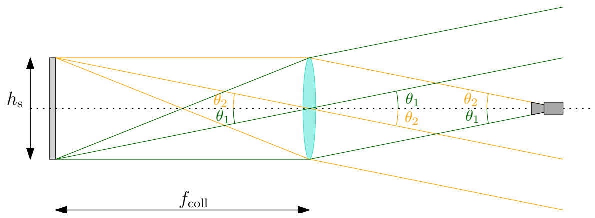


Figure 3.23: Optical configuration of the components of the TinyV3RSE facility.

2. **The high-resolution screen**, whose orientation is set to ensure that the screen and the optical plane of the camera are parallel;
3. **The collimator** ensures that the light coming from the screen and entering the camera is simulated as coming from infinity (or from a very high distance). The collimator is mounted on an optical support that can rotate, change elevation, and be finely adjusted laterally and transversely.

These three modules are visible in the CAD model in Figure 3.24 and in the real image of the facility in Figure 3.25.

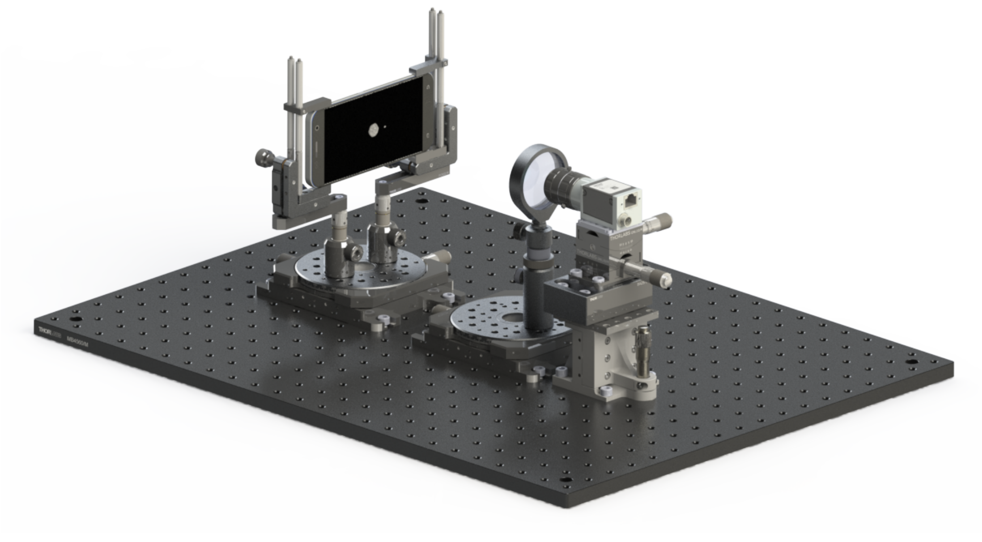


Figure 3.24: CAD model of the TinyV3RSE facility.

Of the three elements, the screen is the only fixed one, while the camera-collimator setup may change depending on the camera properties and as for the considerations performed in Section 3.2.1.

3.2.2.1 Camera

TinyV3RSE has been used with various cameras, depending on the specific project's needs. However, the most used one has currently been the Basler acA1300-22gm

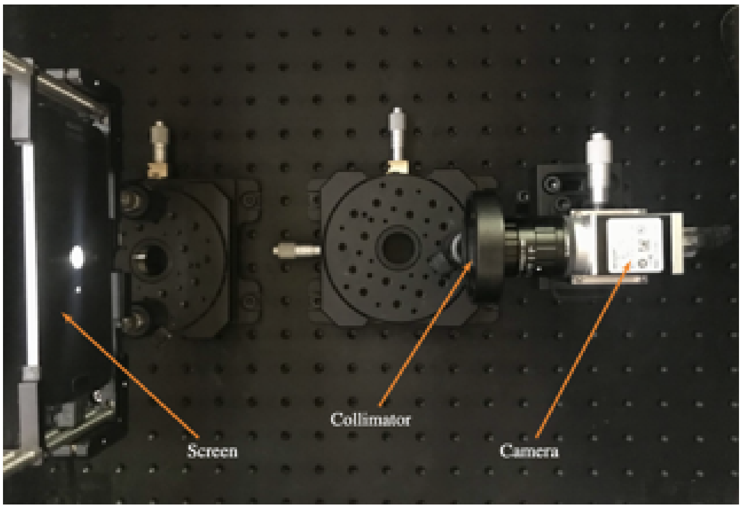


Figure 3.25: Real image of the TinyV3RSE facility.

(CS-Mount)[22] with a 12mm C series fixed focal length lens[23]. The key data-sheet characteristics of the camera assembly are a focal length of *12* mm, a resolution of *1280* pixels $\times$ *960* pixels, a pixel size of *3.75* μm $\times$ *3.75* μm, and a sensor size of *4.9* mm $\times$ 3.6 mm. The camera FOV is $22.6^\circ \times 17^\circ$. The camera is mounted on a dedicated assembly composed of three parts. The first one is a vertical translation stage ensuring vertical assembly control. The second one is a goniometer enabling pitch and roll. The third and last part is a custom mounting adapter to interface between the camera and the optical assembly. In Figure 3.26, it is possible to see a close-up view of the camera assembly pointed towards the collimator. As explained in Section 3.2.1, the distance between the collimator and the camera is kept as small as possible.

3.2.2.2 Screen

The screen is represented by a Galaxy S7 smartphone[24] with a resolution of *2560* pixels $\times$ *1440* pixels, a pixel size of *44.1* μm $\times$ *44.1* μm and a screen size of *112.9* mm $\times$ *63.5* mm. As for the camera and collimator, the screen is mounted on a dedicated assembly composed of two parts. The first is a translational stage, enabling movements of the screen on a plane parallel to the optical one. The second one is a screen holding mechanism that enables re-orientation by changing four pins disposed close to the screen's corners. The choice to use a commercial smartphone as a screen presents several advantages. First, regarding the sizing of the facility,

[22]https://www.baslerweb.com/en/products/cameras/area-scan-cameras/ace/aca1300-22gm-cs-mount/, last time accessed: August 23th 2023

[23]https://www.edmundoptics.com/p/12mm-c-series-fixed-focal-length-lens/14949/, last time accessed August 23th 2023

[24]https://www.displaymate.com/Galaxy_S7_ShootOut_1.htm, last time accessed August 23th 2023

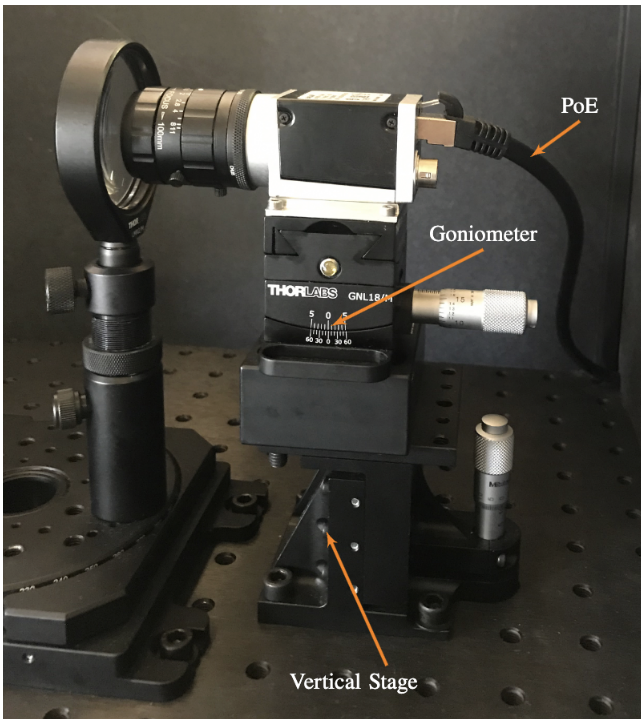


Figure 3.26: Close-up view of the camera and collimator assemblies.

having a compact, high-resolution screen makes it possible to position it within a limited distance from the camera-collimator assemblies, thus ultimately ensuring a compact facility. This is an advantage both in terms of laboratory space and portability and eventual external testing since the optical test bench could easily be moved as carry-on luggage to a different location. Having a smartphone as a screen also makes it simple to set up the interfaces with the server. The smartphone is, therefore, a commercial, hence low-cost solution, which also possesses interesting properties in terms of image contrast. The OLED screen does not suffer from screen bleeding phenomena typical of Liquid Crystal Displays (LCDs) and exhibits a high contrast between inactive and active pixels. This is of particular interest in rendering the pitch-black background of a celestial scene before considering camera noise. The smartphone as a screen solution also exhibits a drawback, given by the screen resolution. Some facilities designed in the past seem to abide by an empirical sampling law for which each pixel of the sensor is to be stimulated at least by four pixels of the screen (or 1:2 if considered linear) [40, 121]. This is to ensure a continuous representation of the environment to the sensor and to satisfy the Nyquist sampling theorem. However, at the time of the facility design, a higher resolution screen than the one considered had not been identified from existing commercial smartphones. The current setup has roughly a 1:1.43 ratio between sensor and screen pixels. It is also observed that this phenomenon did not seem to have played an important disturbance in the performance of the algorithms tested.

3.2.2.3 Collimator

When using a camera with characteristics similar to the Basler acA1300-22gm camera, the collimator used is a 2" diameter N-BK7 plano-convex lens (AR Coating: 350 - 700 nm)[25] with a focal length of *200* mm. The collimator is mounted on a dedicated assembly composed of a roto-translational stage and a post holder, which is used to gain vertical alignment between the collimator and the camera.

Considering the screen and camera characteristics and using Equation 3.23, the camera should be placed at *211.7* mm to perfectly fit the screen's vertical dimension with the vertical length of the camera's FOV. Because of that, the collimator has been chosen with a trade-off study among the plano-convex lenses available as off-the-shelf components. The selected one has been chosen to maximize the observed portion of the screen while avoiding vignetting. Note that when an image is displayed on the screen, it has the size of *112.9* mm $\times$ *63.5* mm. By taking out the calculation with the camera FOV and a collimating distance of *200* mm under the hypothesis of perfectly aligned optical components, a coarse estimation gives that only *80* mm $\times$ *60* mm of the screen is covered by the facility camera FOV. Thus, the image taken by the facility camera is just a portion of the image displayed on the screen. This must be considered when operating and calibrating TinyV3RSE.

The characteristics of the collimator are tightly coupled with the ones of the camera; thus, the design can be easily adopted by using Equation 3.23 depending on the setup considered.

3.2.3 Functional workflow

In terms of interfaces, the screen and server are directly connected to a power outlet while the camera is exchanging data and power via a Power over Ethernet (PoE) cable. Virtual scenes are rendered on the server using CORTO and then sent to the screen via a Wi-Fi or USB. The server is also responsible for activating the camera, receiving and eventually processing the preferred IP algorithm for the images obtained. The screen, collimator, and camera are all enclosed in a box that is closed during the collection of the images. This is done to ensure that proper illumination conditions are met and that artifacts are not introduced on the screen due to external conditions such as reflections, exterior lighting, or shadows of personnel working next to the facility.

In Figure 3.27, the functional workflow used to generate images with the facility is illustrated. The starting point is the simulated world, in which the physical and geometrical properties of the celestial bodies of interest are simulated in the virtual environment in CORTO. Choosing a rendering software to do so is convenient since it enables sampling of such a virtual environment, assuming a particular camera model is positioned from a specific point of view. A rendering of a scene can thus

[25]https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=3279, last time accessed August 23th 2023

be seen as a sampling of this simulated synthetic environment through the physical model of the camera. As it is possible to see from 3.27, this is done twice for any given camera position: The first time to generate an image representative of what the mission camera would be seeing and the second time to get the scene to be projected on the screen at the proper resolution. These are, respectively, the "Ideal Mission Image" and "Screen Image" illustrated in 3.27. Once the screen image is projected, it will stimulate the collimator and camera, which will capture this scene with the real sensor properly positioned, given that a successful calibration ensures the correct alignment of all the components of TinyV3RSE. A final step is required to transform the image captured with the sensor in the facility to an equivalent version of the one caught in the virtual environment. This step is fundamental since, apart from calibration errors, these two images should be geometrically equivalent yet photometrically different. The synthetic one has been generated with an ideal camera model, with no noise and perfect environmental conditions, while the image from the facility encompasses noise and all phenomena typical of a sensor reading. The difference between these two images also represents the same domain gap between real and synthetic images, which TinyV3RSE aims to reproduce for validation.

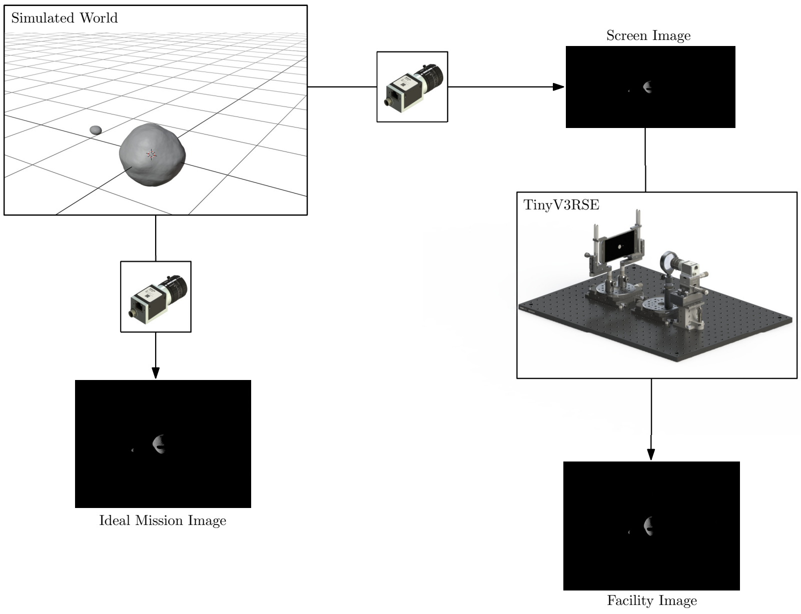


Figure 3.27: Functional workflow in TinyV3RSE.

3.2.4 Calibration

Before using TinyV3RSE, it is essential to perform several calibration procedures to ensure all the components are aligned and performing with the desired accuracy. A thorough review of the most sophisticated calibration in TinyV3RSE is illustrated in [98].

First, a calibration is required to find the intrinsic camera matrix of the equivalent pinhole model for the camera mounted in the facility [122]. Second, the calibration is necessary to take into account the camera-collimator distortion introduced by the lenses. Radial and tangential distortions follow the standard approach in [123]. Other representations can capture distortion, losing the physical interpretation, such as in [124–126]. Lastly, calibration is necessary to estimate the misalignment among the components in the facility. Significant angular errors, if not quantified, would be reflected in the performance of the IP algorithms, nullifying the effort of a validation facility such as TinyV3RSE.

These problems are addressed in a sequential calibration procedure. First, the camera mounted on the facility is calibrated with the algorithm proposed in [127] to find the equivalent pinhole camera model. Then, the alignment of the screen with respect to the camera assembly is estimated by displaying on the screen a series of checkerboards with different orientations, as shown in Figure 3.28, following a similar procedure illustrated in [126]. Through additional cross-hair and other useful patterns, the camera-collimator-screen set is centered and aligned via refinements of the optomechanical elements of TinyV3RSE.

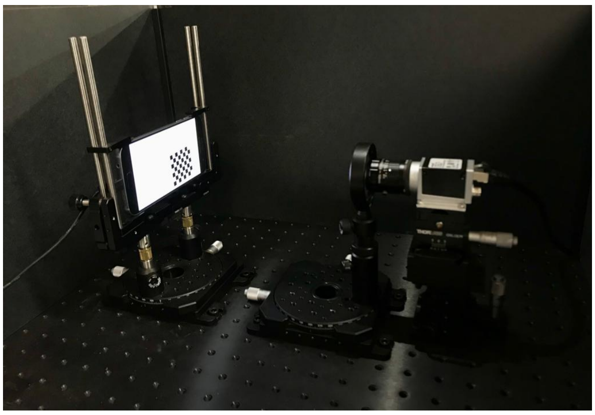


Figure 3.28: A picture of the TinyV3RSE test bench during calibration while using a checkerboard pattern.

At the end of the procedure, the camera intrinsic matrix, the radial and tangential distortion coefficients, and the facility misalignment are estimated. A more detailed description of the most up-to-date calibration procedures used in TinyV3RSE is found in [98, 118]. Finally, It is remarked that at the current stage, photometric calibration is not performed in TinyV3RSE.

3.3 Hardware-in-the-loop with a terrain analog facility

Physical setups such as one that uses 3D printed shape models within robotic facilities or terrain analog resembling the actual environment are valuable for HIL experiments. A variety of these facilities exists [66, 128–131] and are currently operated around the world at various capacities to validate IP and vision-based algorithms. Their main advantages lie in the accurate photometric acquisitions made possible by the natural scattering of the light from an analog material stimulated by the artificial lamps and a real-time simulation framework (no waiting time for rendering the scene). Their main drawback is the high operational costs, the large resources needed to be invested in the design and maintenance, and the rigid setup, which poses geometrical constraints (self-occlusions, self-shadowing, range of the simulation limited to the size of the facility, lighting conditions).

For these motivations, these facilities are often used only in the final stages of an incremental validation campaign. De facto, data acquired with this setting is rarely available outside of the environment, with few notable exceptions of open-access datasets [69, 132].

The capability to accurately detect surface morphological features such as craters and boulders at different scales on the surface of small bodies is of paramount importance for a variety of vision-based applications around small bodies, such as the ones presented in Chapter 4 and Chapter 5. The development of this capability, however, is hindered by significant challenges: the environmental conditions due to the irregularity of the bodies, properties, and distribution of the features, rapidly changing illumination conditions, and most importantly, the lack of publicly available datasets for training, validation, or testing.

This latter challenge is addressed by the tools illustrated in the previous sections, and it is further augmented in this one with the detailed description of a setup used at Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI) to create a general-purpose open-access dataset designed to simplify the access to labeled data about small bodies.

3.3.1 Facility

The Robotic Innovation Center (RIC), part of DFKI, operates a terrain analog facility to test free-climbing robotic systems and demonstrates their mobilities. The facility is roughly $15.0m \times 10.3m$ wide and $4.4m$ high, with a total testing area of about $146.5m^2$. A curtain can be mechanically lowered from the ceiling to make a dark environment representative of space conditions. A studio lamp can then be used within the facility to simulate the Sun's illumination conditions. A sizeable unused volume exists on top of the facility, given that there is a distance of about 12 m between the top of the facility and the ceiling. The surface has been built using real data from the south polar craters on the Moon and images from the Apollo missions and is composed of three continuous paths with 25°, 35°, and 45° slopes and two flat areas at the top and at the bottom. Obstacles such as boulders can be attached at predisposed screw points distributed at regular intervals on the terrain to provide challenging surface conditions for locomotion systems. Finally, a sandy substrate is also simulated with fine-grained material (<1 mm) in the lower part of the facility. A Vicon tracking system [26] is mounted surrounding the facility from above. Finally, in the lower section, a dedicated space serves as a control center containing Vicon's servers and a working station. A 2D map of the facility seen from above is visible in Figure 3.30 while some examples of the views of the terrain are visible in Figure 3.29.

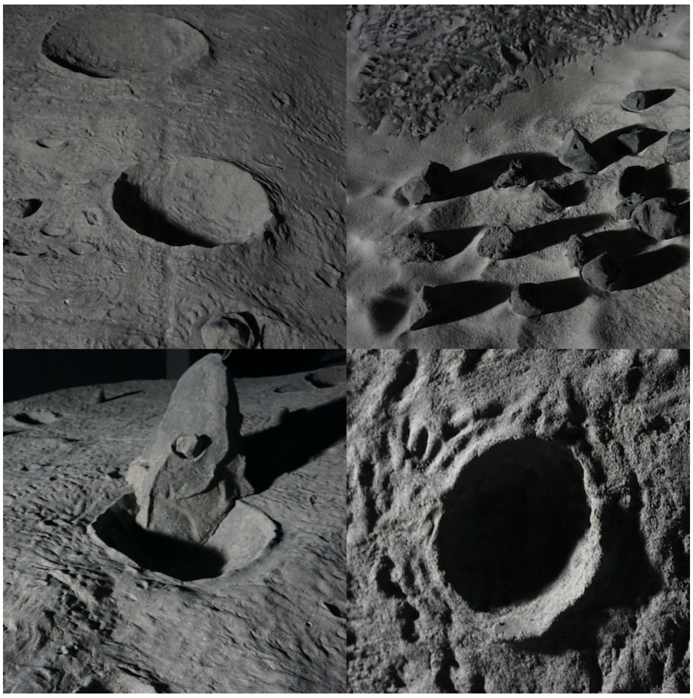
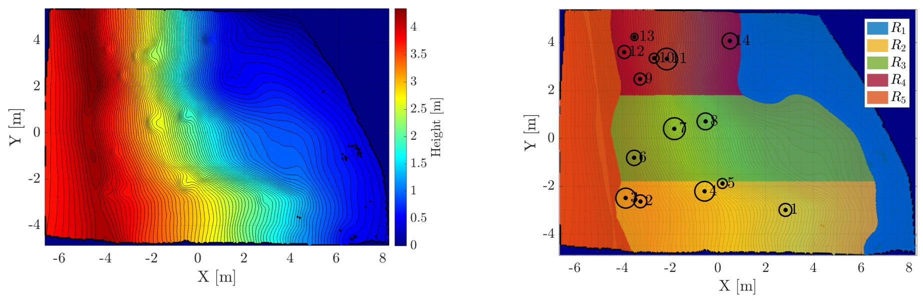


Figure 3.29: Example of views of the facility with simulated lighting conditions.

Usually, the facility simulates surface interactions to test the robotic systems developed at RIC. However, as part of a collaboration with DFKI, it has been exploited in a dedicated activity for visual-based applications.

For simplicity, as illustrated in Figure 3.30(b), the facility is divided into five different regions whose boundaries are designed to separate the three sloped areas at the center and the flat ones at the top and bottom. From R_1 to R_5 each region corresponds to a surface area of roughly $26.7m^2$, $34.0m^2$, $39.1m^2$, $22.5m^2$, and $25.9m^2$. Within these regions, 14 large craters have been manually identified and assigned an ID for referencing. Additionally, for this activity, a boulder field has been created with spare boulders on the lower portion of the terrain (in region R_1).

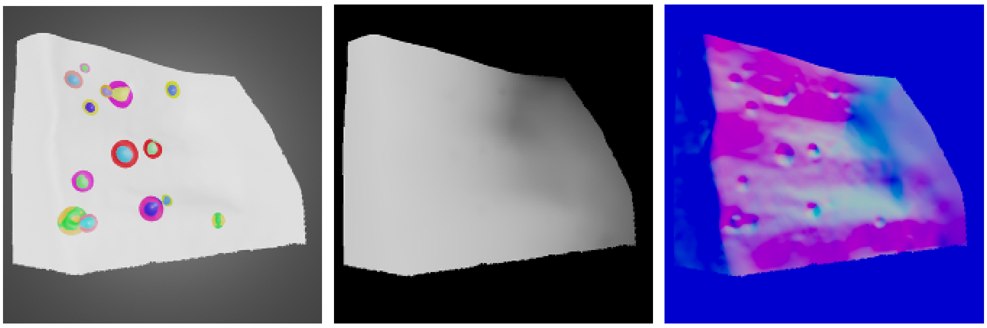
[26] https://www.vicon.com/, last accessed 12th of September 2023.



(a) Heightmap of the facility. **(b)** Naming convention.

Figure 3.30: Heightmap of the facility (a) with isolines (plotted at every *0.05* m intervals). Representation of the naming convention for the regions and large craters of the facility (b) used in this section.

Since a digital terrain model of the facility exists, it is possible to develop a digital twin of the real analog terrain in Blender using CORTO. This is crucial to create an artificial environment in which it is possible to generate artificial labels that go beyond the reconstructed pose provided by the Vicon system. Some possible labels are illustrated in Figure 3.31. Several calibration procedures are necessary to link the real facility with the artificial one, as illustrated in Section 3.3.4.



(a) Craters. **(b)** Depth. **(c)** Slopes.

Figure 3.31: Examples of labels obtained with the artificial environment of the terrain analog facility: (a) semantic segmentation labels for craters, (b) depth map, (b) slope map.

3.3.2 Data collection setup

Exploiting the current design of the facility at the RIC and the free available space above the terrain, a drone is used as an image-collecting device to generate the dataset samples. A commercial drone is a cost-effective option for positioning a camera across the facility without installing other complex equipment. The complete setup used to generate the dataset is illustrated in Figure 3.32.

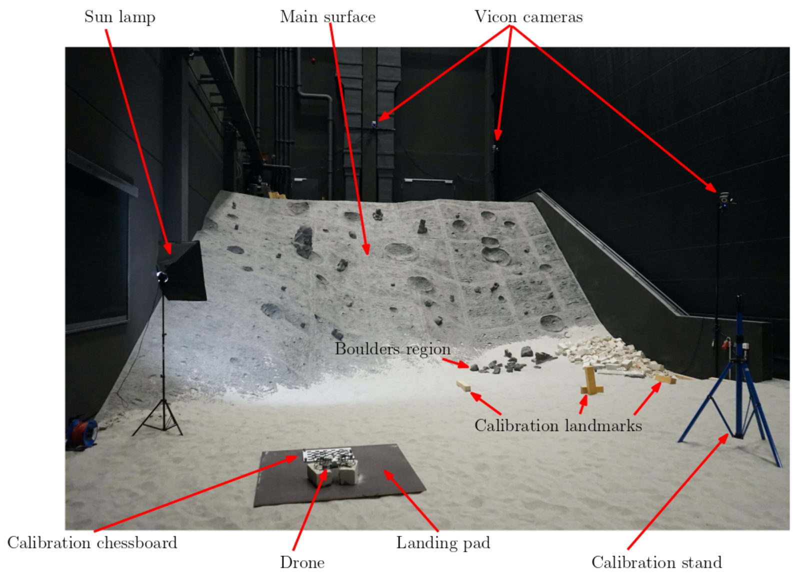


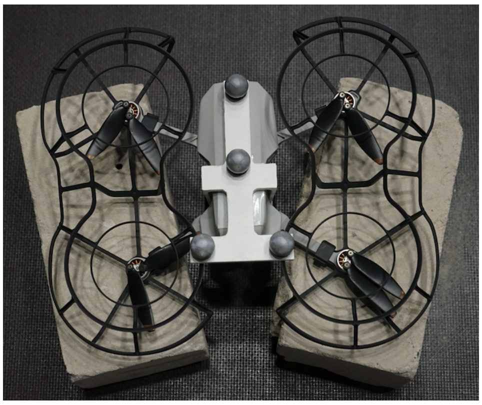
Figure 3.32: Schematic of the setup used in the facility to generate the dataset.

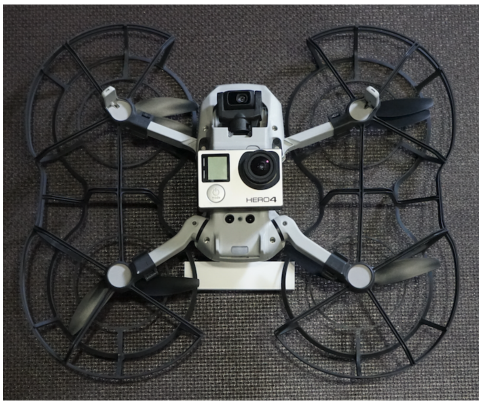
The setup uses the facility and other additional components:

- **Drone**: A commercial drone (The DJI-mini SE [27]) is used as a tool to position cameras around the facility. The drone is flown manually through a dedicated controller and is equipped with the following components, as illustrated in Figure 3.33:
  - **Propeller guards**: They guarantee safe operations and avoid damage to the curtain, the Vicon cameras, or any other element within the facility. As a drawback, the guards lower the drone performance in terms of battery time, shortening the usable flight time with each battery.
  - **3D-printed Vicon tracker stand**: A rigid 3D-printed support structure is attached to the drone for positioning four Vicon markers to allow the detection and tracking of the drone within the facility [28]. This 3D-printed structure demonstrated essential to increase the visibility of the trackers to the Vicon cameras, increasing the number of poses correctly generated within the facility during each flight.
  - **GoPro Hero-4**: A GoPro camera is rigidly attached to the structure of the drone (on the bottom part) to acquire videos with nadir pointing during each flight. By default, the drone is also equipped with a gimbal-stabilized camera. Both cameras can record RGB videos at different framerates, resolutions, and FOV.

[27] https://www.dji.com/id/mini-se/specs, last accessed 11th of September, 2023.
[28] The author would like to personally thank Houssemeddine Jebali at DFKI for having modeled and printed the support for the markers.

- **Calibration chessboard**: A rigid 7×10 calibration chessboard with 35 mm squares and Vicon markers attached to it.
- **Sun lamp**: A studio lamp to qualitatively simulate illumination conditions from the Sun.
- **Vicon system**: A Vicon motion tracking system with cameras surrounding the facility from above.
- **Calibration stand**: A tripod supporting the drone during the calibration procedures.
- **Calibration landmarks**: Visual landmarks used to perform manual calibration in the facility with the Vicon system, the drone, and the calibration stand.
- **Boulders region**: portion of the facility where a boulder field is artificially simulated specifically for this activity. The boulder field is created with spare boulders not currently mounted within the terrain.
- **Main surface**: the main portion of the facility with craters, boulders, slopes, and regions with sand.
- **Take-off pad**: safe region for drone take-off and landing during each flight.


(a) Top view of the Drone.



(b) Bottom view of the Drone.

Figure 3.33: Top (a) and bottom (b) view of the drone used to generate the dataset.

3.3.3 Data generation

With 18 flights of the drone within the facility, the dataset is generated with varying camera properties and illumination conditions. Each flight is executed manually following a predefined flight path. First, the drone traverses the R_2 region across the X axis until arriving at the R_5 region. Then, the drone is moved across the R_5 region, spanning the $R_4 - R_3 - R_2$ region from top to bottom until the boulder field in R_1 is reached. The drone is then moved in a horizontal-vertical cross pattern to mimic a descending trajectory. After this passage, the drone is moved once again over the $R_2 - R_3 - R_4$ region with a random motion. The duration of each flight is approximately five minutes from take-off to landing. Figure 3.34 illustrates all the reconstructed positions of the drone during each flight. Note that the drone is flown visually from the control center within the facility during each flight. This is due to the absence of a GPS signal within the building, making it impossible with the current commercial setting to program any predefined path planning algorithm to automatize flight operations.

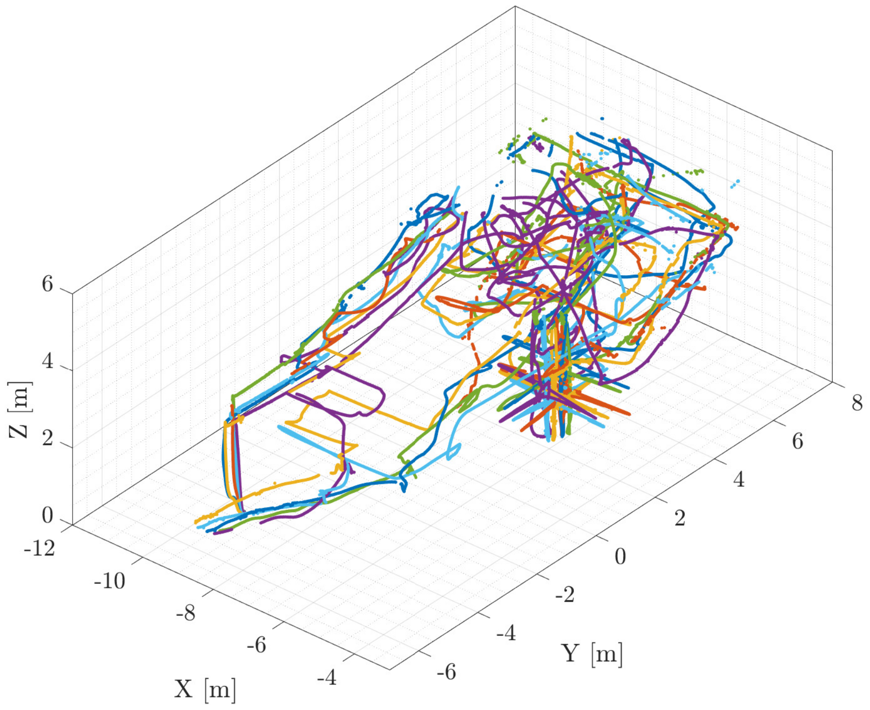


Figure 3.34: Visualization of all the 18 flights used to generate the dataset.

During each flight, video streams, log files from the DJI onboard software, and Vicon data are recorded as raw data. In particular, two streams of videos are recorded simultaneously: one with the integrated DJI camera of the drone and another one with the GoPro rigidly mounted under the bottom structure of the same. The GoPro is considered the primary data-collection sensor. In contrast, the DJI camera is regarded as an opportunistic sensor, given an issue encountered with the self-stabilized gimbal of the DJI camera, which does not make it possible to perform a rigid hand-eye extrinsic calibration, as it is illustrated in subsection 3.3.4.

Each of the flights is executed with different camera settings (resolution,Frames Per Second (FPS), and FOV) and illumination conditions. The camera settings used to generate the videos are summarized in Table 3.5. The first three (A, B, and C) represent the settings used for the GoPro, while the latter two (D and E) are the settings used for the DJI sensor. For each of the GoPro settings, six flights

Table 3.5: Characteristics of the different camera settings used to generate the dataset.

ID	Resolution [px]	FPS	FOV [deg]
A	1920×1080	120	Wide (170)
B	2704×1520	50	Medium (127)
C	1920×1080	120	Narrow (90)
D	1920×1080	50	Wide (83)
E	2720×1530	50	Wide (83)

are executed, varying the illumination conditions and positioning the Sun lamp in different locations around the facility (the lamp is positioned three times in R_1 with a left-center-right configuration pointing towards the center of the facility and then three times in R_5 with the same configuration), generating a total of 18 flights.

3.3.4 Calibration

Several calibration procedures are necessary to correctly collect the samples for the dataset, some of which are performed exclusively for this activity. The Vicon system is assumed to be calibrated; thus, its calibration procedure is not presented.

To link the digital model of the facility with the real one, the reference frame of the Vicon system is exploited as an intermediary. Three trackers are positioned at prominent features in the real facility while the Vicon system records their positions. The positions of the three trackers are then matched in the digital model of the facility. The transformation from the facility and Vicon reference frames can then be easily reconstructed. Since this is a rigid transformation, the calibration is performed only once. This procedure is referred to as the facility-facility calibration since it allows to transform of a position in the real facility (recorded with the Vicon) into a position in the digital facility.

Several data streams are collected from different systems and sensors during each flight. The video streams, the Vicon data, and the flight log data internally recorded by the DJI software must be synchronized. This procedure is referred to as time synchronization and is a laborious task that is performed in a semi-automatic way. To achieve time synchronization, a particular abrupt relative motion needs to be performed during each data-collection event (typically at the beginning) between the drone, the chessboard pattern, and the Vicon system. This is necessary to generate a noticeable and time-limited event that can be recognized by the corners of the chessboard extracted from the camera videos and the poses of the drone recorded with the Vicon. During each flight, this is performed by a sudden sharp movement across the X direction in the facility above the calibration chessboard shortly after take off. During calibration (when the drone is kept stationary), this is achieved by manually moving the calibration pattern in a detectable way both by the drone cameras and the Vicon system.

Currently, the time synchronization is performed only between the video streams and the Vicon system. Also, since these operate at different FPS (100 for the Vicon system, 50 or 120 for the GoPro, and 50 for the DJI camera), the time synchronization accuracy varies depending on the combinations of the three elements. The time synchronization is performed manually by determining the shared intervals in which all three data streams detect the abrupt motion and then allowing a window of frames around this event to be considered as a candidate solution for the synchronization. The proper synchronization is determined as the one achieving the smallest error with the extrinsic calibration procedure illustrated hereafter.

The extrinsic calibration, also referred to as hand-eye calibration, is necessary to find the two unknown transformations. During this calibration, the drone is positioned vertically on the calibration stand within the facility, which allows simple tracking of its pose by the Vicon system. The chessboard calibration pattern is then manually moved within the FOV of the drone's camera, ensuring the Vicon system correctly tracks its markers. During this procedure, the drone's camera performs *n* acquisitions of the chessboard pattern, which should be detected by its *m* corner points of the checkerboard pattern of the calibration chessboard.

During such procedure, the Vicon system, referred to as V, tracks drone C and chessboard CH markers. The purpose of the calibration is to find the set of unknown transformations T_C^D and T_{CH2}^{CH}. T_C^D represents the $\mathcal{CAM}$ reference frame in the $\mathcal{DRONE}$ reference fram. T_{CH2}^{CH} represents the transformation between the $\mathcal{CH}\in$ frame representing the 2D printed chessboard expressed in the $\mathcal{CH}$ reference frame. Since both T_C^D and T_{CH2}^{CH} are rigid transformations, once they are found in the extrinsic calibration procedure, they are assumed constant during each flight. This assumption is not true for the DJI camera of the drone, whose gimbal motion makes the transformation time-dependent. To solve for these two unknown transformations, the coordinates of a point X_i in $\mathcal{CAM}$ are expressed in the calibration setup as:

$$X_i = T_D^C \cdot \left(T_D^V\right)^{-1} T_C^V(t) \cdot \left(T_{CH2}^{CH} \cdot p_i\right) \tag{3.25}$$

where p_i represents the homogeneous coordinates of the corners of the chessboard pattern, $T_C^V(t)$ is the only time-varying transformation during calibration, while all others are rigid transformations. Using the coefficients extracted from the intrinsic calibration about tangential and radial distortions, the point X_i are deformed as:

$$p_j = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot X_i \tag{3.26}$$

$$\begin{aligned} x &= \tfrac{p_{j,1}}{p_{j,3}} \\ y &= \tfrac{p_{j,2}}{p_{j,3}} \\ r^2 &= x^2 + y^2 \end{aligned} \tag{3.27}$$

$$
\begin{aligned}
x' &= 2p_1xy + p_2\left(r_2 + 2x^2\right) \\
y' &= p_1\left(r_2 + 2y^2\right) + 2p_2xy
\end{aligned}
\tag{3.28}
$$

where x' and y' represent the coordinates of the points after applying tangential distortion.

$$
\begin{aligned}
x'' &= x\left(1 + k_1r_2 + k_2r_2^2 + k_3r_2^3\right) \\
y'' &= y\left(1 + k_1r_2 + k_2r_2^2 + k_3r_2^3\right)
\end{aligned}
\tag{3.29}
$$

where x'' and y'' represent the coordinates of the points after application of radial distortion. The projection of the point on the image, given the sets of transformations described above, can thus finally be expressed with the use of the intrinsic matrix K as:

$$
p_k = K \begin{bmatrix} x' + x'' \\ y' + y'' \\ 1 \end{bmatrix}
\tag{3.30}
$$

The prediction of the points p_k in $\mathcal{UV}$ frame performed with the chain of transformations described above can then be confronted with its detection from the images with a simple edge detection algorithm. The difference between the two is used to compute a metric ε_{ext} which is affected by the two unknown transformations T_{CH2}^{CH} and T_C^D.

$$
\varepsilon_{ext} = \frac{\sum_n \sqrt{\left(\sum_m \left(p_k - p_{ext}\right)^2\right)}}{N}
\tag{3.31}
$$

where p_{ext} are the points extracted from images, N is the number of images used in the calibration procedures, and m is the number of points in the chessboard pattern. Using ε_{ext} as a score function and expressing the two unknown transformations in the previous equation are T_{CH2}^{CH} and T_C^D as a seven elements vector made of origin coordinates and quaternion:

$$
T_C^D \rightarrow v_C^D = \left[x_0\ y_0\ z_0\ q_0\ q_1\ q_2\ q_3\ \right]
\tag{3.32}
$$

$$
T_{CH2}^{CH} \rightarrow v_{CH2}^{CH} = \left[x_0\ y_0\ z_0\ q_0\ q_1\ q_2\ q_3\ \right]
\tag{3.33}
$$

an optimization problem is set in Matlab using fmincon with sequential quadratic programming solver and imposing the equality constraint $q_0^2 + q_1^2 + q_2^2 + q_3^2 - 1 = 0$. At convergence, the optimization generates two estimates for the two transformations used for its corresponding flights until the ε_{ext} metric results in an error of a few pixels.

The extrinsic calibration procedure is performed for each camera setting before each flight. Once the transformations are solved, they express the camera position in the facility reference frame, exploiting the rigid transformation between the Vicon and Facility reference frames. This is fundamental in reconstructing the chain of transformations that allow the positioning of a sensor in Blender to recreate image-label pairs in the artificial environment.

The results of one of the extrinsic calibration procedures are illustrated in Figure 3.35 and Figure 3.36. The mean calibration error achieved for ε_{ext} for the specific calibration illustrated is *2.79* px.

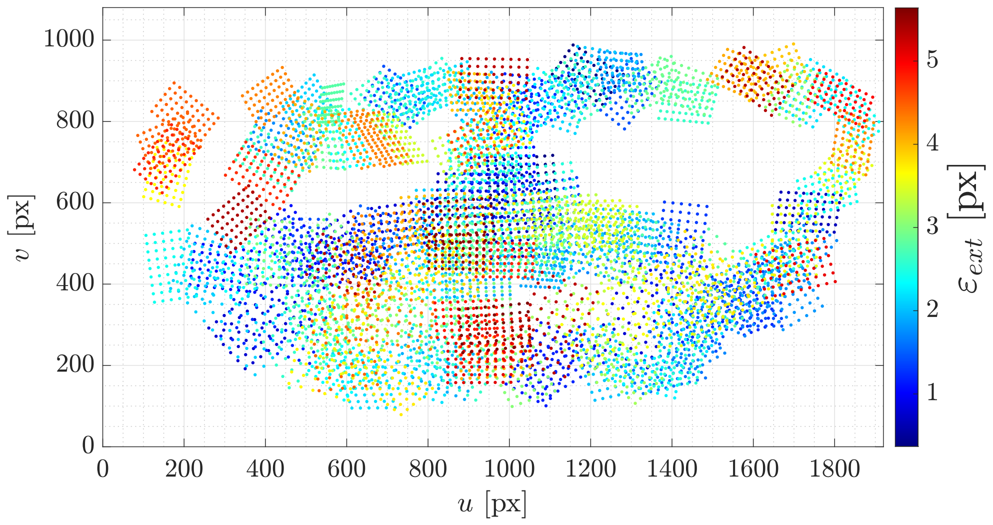
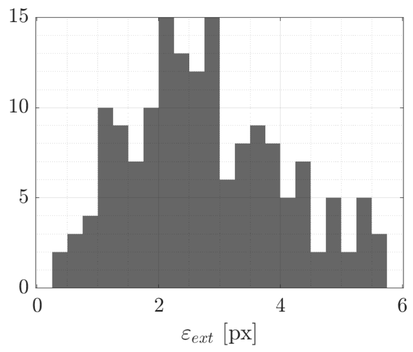
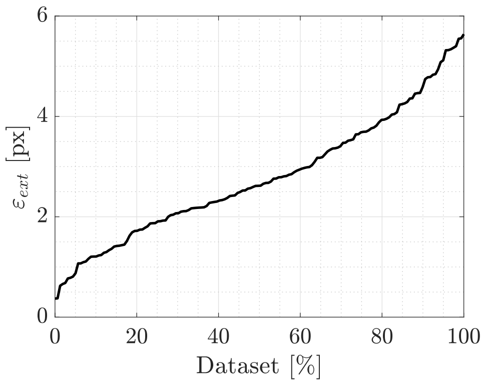


Figure 3.35: Reprojected calibration patterns extracted during the extrinsic calibration procedure for one of the flights. The colors represent the ε_{ext} achieved for each pattern.



(a) Histogram of ε_{ext}.



(b) Cumulative distribution.

Figure 3.36: Histogram (a) and cumulative distribution (b) of ε_{ext} for the extrinsic calibration with the patterns illustrated in Figure 3.35.

Finally, to perform radiometric calibration, black and white images are recorded with the different camera settings in representative illumination conditions encountered during flights. This data can be optionally used to calibrate the level of dark noise and other camera effects in the two sensors used for the data collection.

3.3.5 Preliminary results

The total size of the raw data collected (composed of videos, log files of each flight, and Vicon data) by the *18* flights and the calibration procedures is about *105* Gb. The final dataset will be further processed from this raw data and subdivided into three levels of increasing complexity:

- **Level 0: Images only.** Single images alone or in sequence can be considered at this level to perform a qualitative assessment of the functioning of image processing algorithms with realistic images.
- **Level 1: Images + Poses.** Images are accompanied by full poses reconstructed via the Vicon system and using the transformations estimated using the extrinsic calibration procedure. At this level, the dataset can be used to perform quantitative assessment for visual-based navigation algorithms.
- **Level 2: Images + Poses + Image Labels.** Images are accompanied by fully reconstructed poses and label masks such as the ones illustrated in Figure 3.31. The labels are generated in a digital twin of the facility in which the drone poses are reproduced in Blender using raytracing.

All calibration data will also be available to anyone interested in using the raw data with their personalized calibration algorithms. A random sample of images from one of the flights is illustrated in Figure 3.37

3.4 Final remarks

This chapter illustrated an in-depth overview of the strategies used in this research to generate data for IP algorithms. The absence of data is particularly challenging when considering small bodies for which only a limited statistical sample of observations exists. Three strategies have been illustrated: one using an artificial environment to generate synthetic renderings, one HIL setup exploiting a high-resolution screen in an optical facility, and one HIL setup exploiting a terrain analog facility with slopes, craters, and boulders. What follows is a list of final remarks.

- CORTO is a comprehensive and versatile tool for generating synthetic images of celestial bodies, facilitating the development and validation of image processing and navigation algorithms for space missions. Its capabilities span rendering, noise modeling, hardware-in-the-loop testing, and post-processing, enabling researchers and engineers to simulate realistic scenarios and assess algorithm performance.
- The validation pipeline in CORTO utilizes metrics like normalized cross-correlation and structural similarity via SSIM, ensuring the tool's accuracy and reliability compared to images from previously flown missions.
- CORTO has demonstrated its utility in various case studies and projects, including CubeSat design, lunar missions, and deep learning applications.

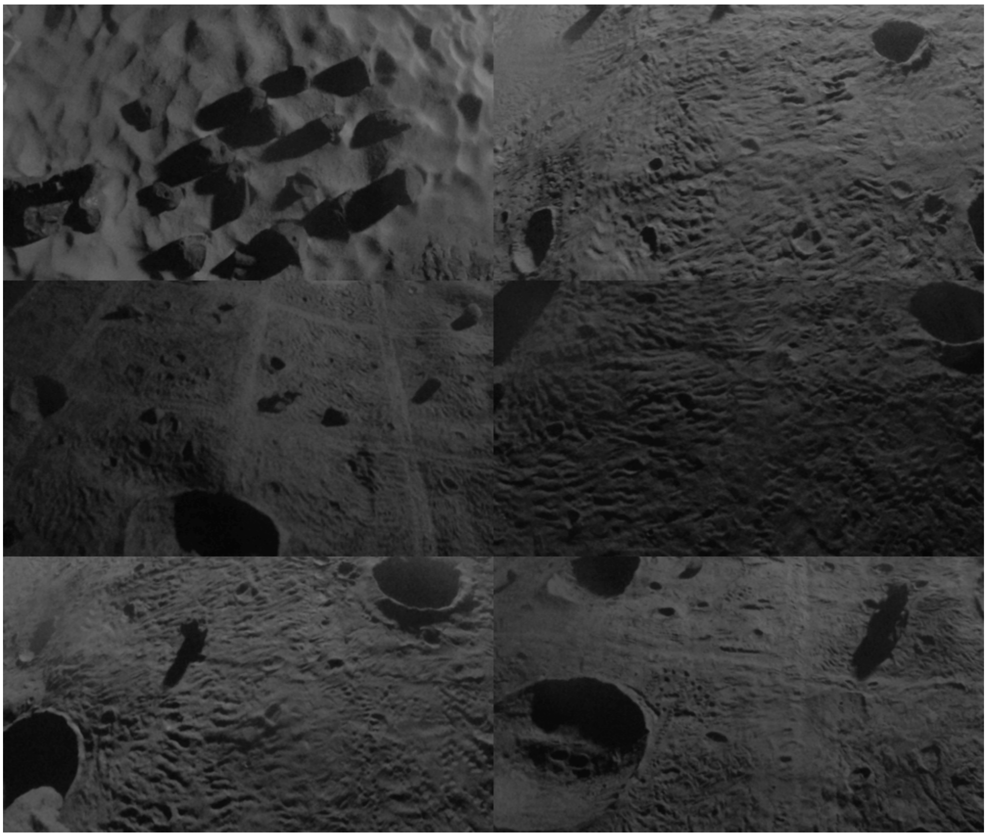


Figure 3.37: Six samples of images captured by the GoPro during one of the flights with ID = B. The brightness of the images is artificially adjusted with $\gamma = 0.7$ for visibility purposes.

- While the tool covers various aspects of celestial body simulation, it can be significantly improved for various applications (use of thermal and infrared sensors, simulation of planets and atmospheric effects, coma effects for comets, and Earth-based applications).
- It is the long-term aim of the CORTO developers to make the tool open-access for any interested researcher to use and develop.
- TinyV3RSE represents a low-cost, versatile, and easy-to-use optical testbench to validate the functioning of IP algorithms with camera effects.
- The terrain analog facility at the RIC in DFKI represents an interesting opportunity to generate a dataset with slopes, craters, boulders, and sandy regions. Although the facility is often used to test the locomotion of robotic systems, it can be easily adapted for visual-based applications.
- A simple drone setup with data-collecting sensors has been implemented to generate a dataset with morphological features typical of small bodies under varying illumination conditions.

- While the required data collection has been successfully performed, its processing into the dataset in level 1 and level 2 is currently under development.
- The drone setup was demonstrated to be highly versatile but with drawbacks. Manually flying the drone in a stable trajectory without a GPS signal proved challenging, especially in the proximity of cratered regions.
- The integrated camera of the drone was demonstrated not to be useful for calibration due to the impossibility of having it rigidly mounted with respect to the Vicon trackers attached to the drone. Future works could consider a dedicated drone designed with a payload bay allocating a variety of sensors (altimeter, lidar, cameras with different FOV) rigidly mounted with respect to the drone, simplifying the calibration procedure and allowing the collection of data from multiple sensors.

4

Segmentation

"Indeed, many movies about artificial intelligence are so divorced from scientific reality that one suspects they are just allegories of completely different concern."

Yuval Noah Harari, *21 Lessons for the 21st Century*

Small bodies are characterized by a variety of shapes with different physical, orbital, composition, and surface properties. These are only roughly observed from ground-based and space-based telescopes and require further close-up investigations from visiting spacecraft.

In particular, surface morphological features such as color variations, craters, and boulders become visible only relatively close to the body. These features could enable crucial autonomous capabilities onboard spacecraft if robustly detected under varying illumination conditions.

Craters and large boulders can be used as landmarks for navigation, as performed with the human-in-the-loop approach described in [60], or for autonomous onboard scientific acquisitions, commanding the spacecraft pointing and acquisition of valuable scientific data based on the appearance of selected features in the images. Autonomous landing systems would benefit from an advanced comprehension of the touchdown site. Detecting slopes, cratered regions, and large boulders would prove crucial in performing real-time hazard estimation, potentially providing the capability onboard for autonomous landing site selections.

This chapter addresses the detection of these features as an image segmentation task performed using DL architectures. Semantic segmentation can be defined [22] as the capability to perform both object recognition and accurate boundary segmentation at pixel level. Crucially, semantic segmentation can be considered a transformation of the image content from pixel intensity to class belonging. This transformation is powerful since the segmentation map makes available a more complex source of information for an onboard system to act on.

Many techniques have been developed in the past decades to perform image segmentation with small bodies, most notably using traditional IP methods [53] during flybys and DL ones [24, 49, 54–59], as already discussed in Section 2.1.2. The aforementioned works, however, showcased three major inconveniences that are addressed in this chapter.

First, they do not fully appreciate the complexity of the classes of features existing on a small-body surface, often not considering more than three meaningful layers or focusing only on safe-unsafe pixel classification. Second, most do not include uncertainty quantification metrics, which could be paramount for a real operational scenario, as illustrated in [133]. Third, they often require extensive manual preparation of the data since (apart from [49, 55, 133] which use digital terrain maps and [58] which actively exploits ray-tracing capabilities) large, realistic, annotated datasets are rarely available. This is a significant deficiency for high-performing data-driven methods demanding a large amount of annotated data to produce reliable architectures capable of working in real mission scenarios.

In this chapter, image segmentation applications with DL architectures are illustrated. First, the problem is framed as a multi-layer segmentation task to distinguish between several morphological features. Then, the focus is shifted to the robust segmentation of boulders scattered across the surface of small bodies, to be used mainly for navigation purposes. Finally, a comprehensive, multi-purpose image segmentation dataset is illustrated that combines the lessons learned from the design of these architectures.

4.1 Multi-layer segmentation

Small bodies exhibit a variety of surface morphological features that, being difficult to extract from images, are often not considered. In this section, UNet architectures are used to generate predicted masks over the surface of small bodies that distinguish between five different taxonomic classes of morphological features: background, surface, crater, boulder, and terminator region. Figure 4.1 shows an example of this mask.

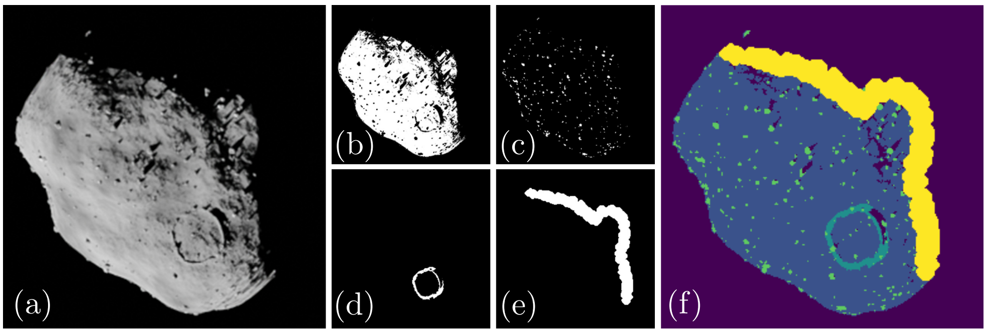


Figure 4.1: Complete image of the small-body model of Lutetia (a), surface(b), boulders(c), craters (d), and terminator region(e) layers and true segmentation mask (f).

The masks are obtained in CORTO exploiting the ray-tracing of the Cycles rendering engine in Blender, assigning different pass indices to the different layers of a model, as illustrated in detail in [101] and in Section A.1.1.

Exploiting the automatic labeling capability of an artificial environment in Blender, a dataset of segmentation masks referred to as $\mathcal{DS}_1$ is generated for various bodies. More details about this dataset can be found in the appendix in Section A.1.1. $\mathcal{DS}_1$ is composed of five sets: four made of synthetic images generated in CORTO ($\mathcal{DS}_1^{D-1}$, $\mathcal{DS}_1^{D-2a}$, $\mathcal{DS}_1^{D-2b}$, and $\mathcal{DS}_1^{D-3}$), and one made of real images that have been manually labeled ($\mathcal{DS}_1^{D-4}$). $\mathcal{DS}_1^{D-1}$ is used for training, validation, and testing and is made of image-mask pairs of seven different bodies. $\mathcal{DS}_1^{D-2a}$, $\mathcal{DS}_1^{D-2b}$, and $\mathcal{DS}_1^{D-3}$ are only used in inference and are made using two bodies that have never been seen during training. $\mathcal{DS}_1^{D-3}$ is the only dataset that simulates an acquisition during a flyby trajectory, assessing the segmentation capabilities in such a scenario. Finally, $\mathcal{DS}_1^{D-4}$ is made of a small set of images manually labeled by the author in [101].

4.1.1 Convolutional architectures

To perform multi-layer image segmentation, a CNN architecture based on the UNet structure is adopted (See Section 2.2.2).

First, the encoding layers of the UNet are trained as a conventional CNN for a classification task. The task is formulated as the work in [103], explicitly developing an encoder capable of extracting features of interest from small-body images. The classes correspond to the names of the target bodies, thus linking the body's appearance (and its surface features) with its class. The network input is a grayscale image of the target body, while the output is a vector of *7* elements, each representing the *softmax* probability of the image belonging to that specific class.

The architecture of the encoder is illustrated in Figure 2.8(a), its design is summarized in Table 4.1 while the hyperparameters considered during training are reported in Table 4.2. Once the encoder has been trained, it is embedded into the UNet for the final image segmentation task.

A thorough hyperparameter search based on a refined search-grid approach determines the best network. The Leaky Rectified Linear Unit (Leaky ReLU) is used in all the convolutional layers of the CNN while the Rectified Linear Unit (ReLU) is used in all the layers of the neural network portion of the CNN apart from the last one, the output layer, which uses the *softmax* activation function.

As illustrated in Table 4.1, the CNN architecture is divided into four portions, respectively, from top to bottom: input, convolutional layers, neural networks layers, and output. The results of convolution and activation at each depth of the CNN are copied in the encoding layers of the UNet while stacked in the decoding layers of the same.

The encoder can also be designed with different strategies. For example, in a previous iteration in [134], transfer learning is used on a pre-trained MobileNet-V2 architecture, a well-known flexible architecture [135] that had been previously

Table 4.1: Architecture of the CNN considered as classifier. The total number of parameters is *1,474,951* (5.62 MB), all of which are trainable.

ID	Layer type	Output Shape	Parameters	Connections
I	InputLayer	($\mathcal{B}$, 128, 128, 1)	0	C1
C1	Conv2D	($\mathcal{B}$, 128, 128, 16)	160	LR1
LR1	LeakyReLU	($\mathcal{B}$, 128, 128, 16)	0	P1
P1	MaxPooling2D	($\mathcal{B}$, 64, 64, 32)	0	C2
C2	Conv2D	($\mathcal{B}$, 64, 64, 32)	4640	LR2
LR2	LeakyReLU	($\mathcal{B}$, 64, 64, 32)	0	P2
P2	MaxPooling2D	($\mathcal{B}$, 32, 32, 64)	0	C3
C3	Conv2D	($\mathcal{B}$, 32, 32, 64)	18496	LR3
LR3	LeakyReLU	($\mathcal{B}$, 32, 32, 64)	0	P3
P3	MaxPooling2D	($\mathcal{B}$, 16, 16, 128)	0	C4
C4	Conv2D	($\mathcal{B}$, 16, 16, 128)	73856	LR4
LR4	LeakyReLU	($\mathcal{B}$, 16, 16, 128)	0	P4
P4	MaxPooling2D	($\mathcal{B}$, 8, 8, 256)	0	C5
C5	Conv2D	($\mathcal{B}$, 8, 8, 256)	295168	LR5
LR5	LeakyReLU	($\mathcal{B}$, 8, 8, 256)	0	P5
P5	MaxPooling2D	($\mathcal{B}$, 4, 4, 256)	0	DO1
DO1	Dropout	($\mathcal{B}$, 4, 4, 256)	0	FC
FC	Flatten	($\mathcal{B}$, 4096)	0	D1
D1	Dense	($\mathcal{B}$, 256)	1048832	DO2
DO2	Dropout	($\mathcal{B}$, 256)	0	D2
D2	Dense	($\mathcal{B}$, 128)	32896	O
O	Output	($\mathcal{B}$, 7)	903	

Table 4.2: Hyperparameters of the CNN used for training.

Parameter	Value
Batch size	200
Optimizer	Adam
Activation function	ReLU, LeakyReLU, and Softmax
α parameter of the LeakyReLU	0.3
Convolution kernel size	3x3
Pooling kernel size	2x2
Dropout value (DO1)	0.2
Dropout value (DO2)	0.2
Loss metric	SCCE
Accuracy metric	accuracy
Epochs	100

trained for different tasks and types of images that are entirely different from the one associated with small-bodies. Instead, in [101], the authors experimented with a custom-made encoder to potentially increase the performance and reduce the network's size. The work presented in this section reflects the one performed in [101].

The efficacy and simplicity of the UNet architecture has already been proven both in the broader computer vision domain [136] and in various space applications [49, 55, 58, 133, 134], representing the state-of-the-art approach for image segmentation [22].

The schematic of the UNet architecture used is illustrated in Figure 2.8(b), its design is summarized in Table 4.3, while its hyperparameters are listed in Table 4.4.

Table 4.3: Architecture of the UNet considered in this section. The total number of parameters is *1,225,413* (4.67 MB), *832,613* (3.18 MB) of which are trainable and *392,320* are not.

ID	Layer type	Output Shape	Parameters	Connections
I	InputLayer	($\mathcal{B}$, 128, 128, 1)	0	E1
E1	Encoder	($\mathcal{B}$, 128, 128, 16)	160	E2, CC4
E2	Encoder	($\mathcal{B}$, 64, 64, 32)	4640	E3, CC3
E3	Encoder	($\mathcal{B}$, 32, 32, 64)	18496	E4, CC2
E4	Encoder	($\mathcal{B}$, 16, 16, 128)	73856	E5, CC1
E5	Encoder	($\mathcal{B}$, 8, 8, 256)	295168	UP1
UP1	Sequential	($\mathcal{B}$, 16, 16, 128)	295424	CC1
CC1	Concatenate	($\mathcal{B}$, 16, 16, 256)	0	DO1
DO1	Dropout	($\mathcal{B}$, 16, 16, 256)	0	LR1
LR1	LeakyReLU	($\mathcal{B}$, 16, 16, 256)	0	UP3
UP2	Sequential	($\mathcal{B}$, 32, 32, 64)	147712	CC2
CC2	Concatenate	($\mathcal{B}$, 32, 32, 128)	0	DO2
DO2	Dropout	($\mathcal{B}$, 32, 32, 128)	0	LR2
LR2	LeakyReLU	($\mathcal{B}$, 32, 32, 128)	0	UP4
UP3	Sequential	($\mathcal{B}$, 64, 64, 32)	36992	CC3
CC3	Concatenate	($\mathcal{B}$, 64, 64, 64)	0	DO3
DO3	Dropout	($\mathcal{B}$, 64, 64, 64)	0	LR3
LR3	LeakyReLU	($\mathcal{B}$, 64, 64, 64)	0	UP5
UP4	Sequential	($\mathcal{B}$, 128, 128, 16)	9280	CC4
CC4	Concatenate	($\mathcal{B}$, 128, 128, 32)	0	DO4
DO4	Dropout	($\mathcal{B}$, 128, 128, 64)	0	LR4
LR4	LeakyReLU	($\mathcal{B}$, 128, 128, 128)	0	CT1
CT1	Conv2DTranspose	($\mathcal{B}$, 128, 128, 128)	36992	DO5
DO5	Dropout	($\mathcal{B}$, 128, 128, 128)	0	CT2
CT2	Conv2DTranspose	($\mathcal{B}$, 128, 128, 256)	295168	DO6
DO6	Dropout	($\mathcal{B}$, 128, 128, 256)	0	O
O	Conv2DTranspose	($\mathcal{B}$, 128, 128, 5)	11525	

The architecture in Table 4.3 is divided into five portions, from top to bottom: input, encoder, decoder, head, and output. The encoder is constituted by the frozen convolution layers of the CNN architecture in Table 4.1 while the decoder is generated by stacking such layers with new upsampling layers taken from the pix2pix[29] architecture in TF.

The contracting portion of the network (encoder) is composed of a succession of convolution, Leaky ReLU activation functions, and max-pooling layers, which progressively increase in depth and reduce in size (i.e., height and width). The expansive portion (decoder) is made by a combination of transpose convolution (light-red and light-pink blocks in Figure 2.8(b)), Leaky ReLU, and upsampling layers of reducing depth and increasing size. This symmetric nature is what gives the network its characteristic "U" shape and name. The output of the convolutional layers of the encoder is copied and stacked in the corresponding layers of the decoder, as represented by the blue arrows in Figure 2.8(b). Moreover, the lack of a fully connected layer in the middle is a network characteristic.

The input is represented by a grayscale image, the output by a preliminary $128 \times 128 \times 5$ tensor which is processed as final output to be a 128×128 image whose pixel values span from 0 to 4, each corresponding to a specific layer of the small-body. These are, from 0 to 4: background, surface, crater, boulder, and terminator region. In Figure 4.2 it is possible to see an exploded view of the output portion of the UNet before and after the application of the *argmax* function. Each of the five layers that makes the output tensor is a 128×128 matrix containing unbounded float values. Each pixel of the matrix represents a scalar score for that pixel belonging to that specific layer, the score being visualized with a *jet* colormap in Figure 4.2; the higher the score (red), the higher the chance of that pixel belonging to that class, vice-versa for lower scores (blue). By applying the *argmax* function, the $128 \times 128 \times 5$ tensor is reduced to a 128×128 matrix where each pixel can assume a value from 0 to 4 (i.e., the values representing the morphological classes) so that the output matrix represents the predicted segmentation mask. Note that the colors used for each class, from the *viridis* colormap, which is visible in Figure 4.2, are the same that used for the remainder of this section when defining the multi-layer segmentation masks.

Three variations of the same UNet architecture are investigated, referred to as UNet Synthetic ($\mathrm{UNet_S}$), UNet Synthetic Augmented ($\mathrm{UNet_S^A}$), and UNet Real ($\mathrm{UNet_R}$). The three architectures share the same structure described in Table 4.3, differing only by their weights and biases, which are the results of different training strategies. $\mathrm{UNet_S}$ is trained using $\mathcal{DS}_{1}^{D-1}$, composed only of synthetic images. $\mathrm{UNet_S^A}$ and $\mathrm{UNet_R}$ are trained using $\mathcal{DS}_{1}^{D-4}$, composed only of real images. At initialization of $\mathrm{UNet_S^A}$, transfer learning is performed by sharing the same weights and biases of $\mathrm{UNet_S}$. $\mathrm{UNet_S^A}$ starts its training with knowledge acquired by the architecture with synthetic images only. It uses this knowledge for a new training phase with real image-label pairs. On the contrary, $\mathrm{UNet_R}$ starts with a randomized set of weights and biases, not exploiting the previous training episodes with synthetic images. The choice to keep the three architectures identical is made as a proof of concept to remove architectural differences and to isolate the contribution caused

[29]https://www.tensorflow.org/tutorials/generative/pix2pix, last accessed on 15th of March 2022.

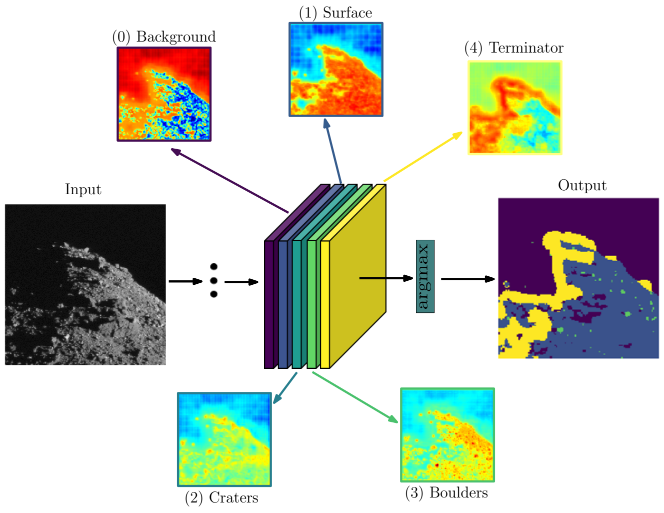


Figure 4.2: Exploded view of the *128* × *128* × *5* output tensor before it is processed to generate the output mask. The color of the layers illustrated in this figure is used to represent all the segmentation masks.

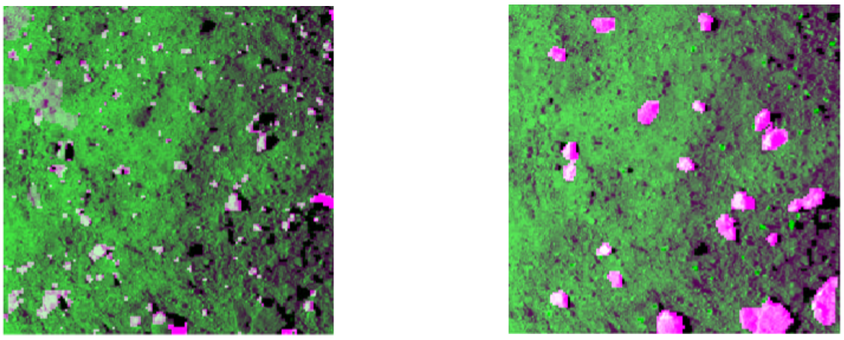
by transfer learning and different training strategies.

As for the CNN classifier, a thorough hyperparameter search based on a refined grid-search approach defines the best three UNet networks. The Leaky ReLU is used in all the convolutional layers, the ReLU is used in all the layers of the neural network portion of the CNN apart from the last one, the output layer, which uses the *softmax* activation function.

Table 4.4: Hyperparameters used in training of the UNet.

Parameter	UNet$_S$	UNet$_S^A$	UNet$_R$
Batch size	50	70	70
Optimizer		Adam	
Activation function		ReLU, LeakyReLU	
α parameter of the LeakyReLU		0.3	
Convolution kernel size		3x3	
Pooling kernel size		2x2	
Dropout values (DO1-DO4)	0.2	0.6	0.1
Dropout values (DO5-DO6)	0.4	0.6	0.1
Weight Background	0.09	0.09	0.09
Weight Surface	0.09	0.09	0.09
Weight Craters	0.45	0.35	0.35
Weight Boulders	0.23	0.33	0.33
Weight Terminator	0.14	0.14	0.14
Loss metric		WSCCE	
Accuracy metric		mIoU	
Epochs	100	1500	1500
Learning rate	*1.3 e*$^{-3}$	*1.0 e*$^{-6}$	*1.0 e*$^{-3}$

Finally, an additional hybrid architecture, referred to as UNet Hybrid (UNet$_H$), making use of mixed prediction between UNets$_S$ and UNet$_S^A$ is investigated. This was prompted by the observation that the automatically-labeled boulders in $\mathcal{DS}_1^{D-1}$ tend to robustify UNets$_S$ in the detection of small-to-medium size boulders (see Figure 4.7), while the manually-labeled ones in $\mathcal{DS}_1^{D-4}$ have a similar effect on UNet$_S^A$ for large boulders (see Figure 4.14), as it is possible to see in Figure 4.3. In an attempt to combine the strength of these two networks, their predictions are combined as illustrated in Figure 4.4.



(a) Detection by UNets$_S$. **(b)** Detection by UNet$_S^A$.

Figure 4.3: Same scene of Bennu surface with an overlay of the boulders detected by UNets$_S$ (a) and UNet$_S^A$ (b).

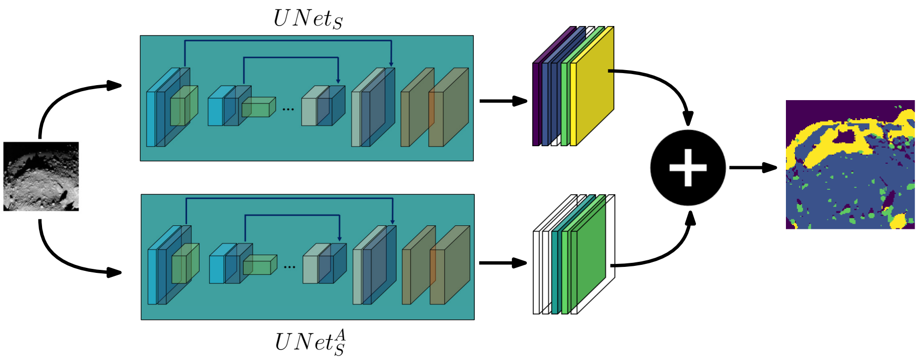


Figure 4.4: Hybrid architecture UNet$_H$ which uses the contribution of multiple networks for a qualitatively more realistic segmentation.

UNet$_H$ is realized by running the same input image twice through the same architecture, instanced by different weights and biases representing UNets$_S$ and UNet$_S^A$. This architecture generates two sets of $128 \times 128 \times 5$ raw outputs for each input image. As illustrated in Figure 4.4, the *1st*, *2nd* and *5th* layers (corresponding respectively to background, surface, and terminator region) are predicted entirely by UNets$_S$. The prediction of the *3rd* layer is performed by UNet$_S^A$, since it seems better at detecting craters from real images. Finally, the *4th* layer, the one predicting the boulders, is computed as a weighted mix between the predictions from UNets$_S$ and UNet$_S^A$ using the following equation:

$$\mathbf{I}_4^{\mathrm{UNet_H}} = log_{10}(\gamma) \cdot \mathbf{I}_4^{\mathrm{UNet_S}} + (1 - log_{10}(\gamma)) \cdot \mathbf{I}_4^{\mathrm{UNet_S^A}} \tag{4.1}$$

where γ is a weighting parameter set to vary between 1 and 10 to mix the contribution of the two predictions. Since no reference ground truth mask is generated for this scenario, its performance is only assessed qualitatively.

4.1.2 Uncertainty quantification

In this section, a preliminary step towards the inclusion of uncertainty quantification in the pixel class prediction by the UNet is attempted. This is done by leveraging prior efforts on the topic from [137], which introduces a methodology to quantify uncertainty from predictive entropy, and the from [133], which showcases how such uncertainty could be operationally used for robust, safe landing-site selection. The approach to generate uncertainty maps is schematized in Figure 4.5.

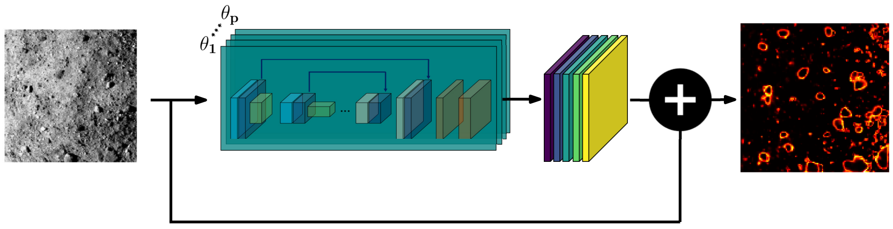


Figure 4.5: Architecture to generate the uncertainty maps.

Predictive entropy can model both aleatoric and epistemic uncertainty [137], the first caused by environmental variability and the second by the model. The approach described in [137] exploits Bayesian inference and the non-deterministic nature of the network architecture (achieved by incorporating dropout) to quantify the uncertainty as predicted entropy. Both phenomena can also be applied to $\mathrm{UNet_S}$, $\mathrm{UNet_S^A}$, and $\mathrm{UNet_R}$.

Considering only one architecture for simplicity, the same instance of weights and biases θ is used, and the input image is run P times without changing them ($P = 20$ in this analysis). Because dropout is extensively used in the architectures, the output to the same image changes at each iteration as the dropout changes the active connections used across the network. Note that the weights and biases are the same at each prediction, but the dropout randomly nullifies some connections; hence θ_p represents the same instance of the network with some connections randomly removed. This produces $P \times 128 \times 128 \times 5$ output tensors, considering the raw outputs before applying the *argmax* function (see Figure 4.2). From these multiple sets of raw outputs, the uncertainty is computed as pixel-wise predictive entropy as [137]:

$$\hat{\mathbb{H}}[\mathbf{y}|\mathbf{x}] = -\sum_{i=1}^{K} \left[\frac{1}{P} \sum_{p=1}^{P} softmax(y_{i,p}|\mathbf{x}, \theta_p) \right] \cdot \log \left[\frac{1}{P} \sum_{p=1}^{P} softmax(y_{i,p}|\mathbf{x}, \theta_p) \right] \tag{4.2}$$

where **x** and **y** are respectively the vector of input and output of the network for each pixel, K is the number of classes over which the predictive entropy is computed ($K = 5$ in this study), p is referred to the p-th sample considered and finally $softmax(y_{i,p}|\mathbf{x}, \theta_p)$ is the probability that the network assigns the pixel to the i-th class during p-th sample given the input **x** and obtained with a set of weights, biases, and dropout combination provided by θ_p.

Nominally, the network prediction is generated using the *argmax* function, as illustrated Figure 2.8(b). However, the softmax function is used instead to generate $\hat{\mathbb{H}}[\mathbf{y}|\mathbf{x}]$. By applying this approach to the prediction of the class of each pixel, an uncertainty map is generated accompanying each segmentation map. The uncertainty map is visualized with an *inferno* colormap (from black representing low uncertainty to yellow representing high uncertainty), as illustrated in Figure 4.5.

The uncertainty is exploited to assign pixels associated with high uncertainty to a 6^{th} additional layer (a sort of extra layer that highlights unstable predictions in the segmentation map that might be avoided). This is implemented easily by setting a global threshold (quantified as a scalar between 0 and 1, which scales between the minimum and maximum values of predicted entropy for each image) and moving the pixels above this threshold into the extra 6^{th} layer of the segmentation map. The validation datasets of $\mathcal{DS}_1^{D-1}$ and $\mathcal{DS}_1^{D-4}$ are used to select an appropriate value for such a threshold. The threshold is selected to maximize the mIoU for all images in the validation sets, which is then used in inference only on images from the test sets. The threshold is found to be equal to 0.91, 0.54, 0.48, and 0.23 respectively for the UNet$_\mathrm{S}$ (on $\mathcal{DS}_1^{D-1}$), UNet$_\mathrm{S}$ (on $\mathcal{DS}_1^{D-4}$), UNet$_\mathrm{S}^\mathrm{A}$ (on $\mathcal{DS}_1^{D-4}$), and UNet$_\mathrm{R}$(on $\mathcal{DS}_1^{D-4}$).

4.1.3 Results

This section illustrates the performance of the segmentation architectures on the test sets of $\mathcal{DS}_1$.

4.1.3.1 Segmentation over $\mathcal{DS}_1^{D-1}$, $\mathcal{DS}_1^{D-2}$, and $\mathcal{DS}_1^{D-3}$

The performance of UNet$_\mathrm{S}$ over the synthetic datasets is summarized in Table 4.5. It is remarked that the mIoU over $\mathcal{DS}_1^{D-1}$ drops from $\sim 60\%$ on the validation set to $\sim 56\%$ on the test set, which is also the highest value of mIoU achieved across all synthetic test sets. Albeit this drop in performance, the network shows good generalization capabilities when it is tested with new models that have never been encountered during training ($\mathcal{DS}_1^{D-2a}$ and $\mathcal{DS}_1^{D-2b}$) and in a flyby scenario with an unknown body($\mathcal{DS}_1^{D-3}$).

Comparing the performance presented in Table 4.5 (that reflects the one in [101]) with the one of a previous iteration in [134], a slight drop is observed. Since both works use the $\mathcal{DS}_1$ dataset, the difference is solely attributed to the networks.

By design, the UNet architecture presented in this section has a considerably smaller encoder (410^5 parameters) than the one used in [134] (about 1.810^6

Table 4.5: Summary of the mIoU of the UNet_S for the different test cases expressed as a percentage.

Test case		Background	Surface	Craters	Boulders	Terminator	Mean
	min	11.11	70.05	0.16	1.08	0.27	
D-1	**mean**	**88.16**	**91.57**	**14.09**	**28.34**	**57.52**	**55.93**
	max	100.00	99.54	100.00	81.06	100.00	
	min	71.36	43.01	0.17	1.16	0.10	
D-2a	**mean**	**95.91**	**91.63**	**10.68**	**19.97**	**44.67**	**52.57**
	max	99.82	97.31	100.00	58.46	100.00	
	min	6.25	71.09	0.03	2.79	0.09	
D-2b	**mean**	**88.05**	**91.60**	**9.20**	**23.35**	**60.50**	**54.54**
	max	100.00	99.18	100.00	59.28	100.00	
	min	30.61	1.11	0.48	2.13	0.41	
D-3	**mean**	**94.08**	**73.63**	**11.91**	**16.53**	**58.19**	**50.87**
	max	100.00	98.88	47.62	50.21	100.00	

parameters). This trend is also observed for the whole architecture. Moreover, the architecture presented in this section is trained from scratch using solely small body images from $\mathcal{DS}_1$. On the contrary, the one in [134] is fine-tuned from a previously trained model on completely different images than the ones typical of small bodies.

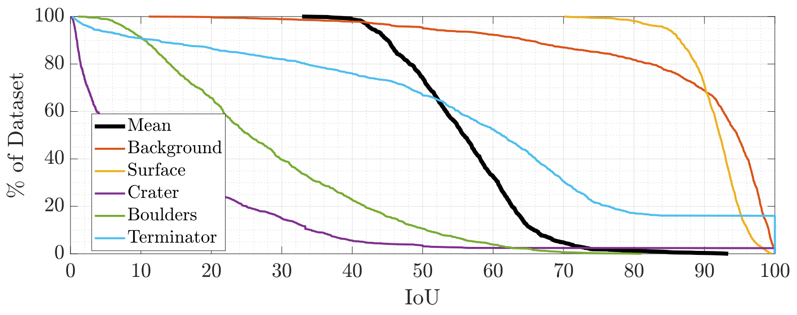
Since the performance is observed to be slightly worse or similar to that in [134], it can be concluded that the encoder's specialization with small body features did not seem to have brought relevant advantages, albeit a smaller architecture has been used.

In Figure 4.6, it is possible to see the cumulative values of IoU and mIoU across all datasets. In particular, it is observed that the capability of the UNet to detect background, surface, and terminator regions accurately boosts the mIoU. On the other hand, the network's worse performances in robustly detecting craters and boulders seem to be the critical element in dragging down the mIoU.

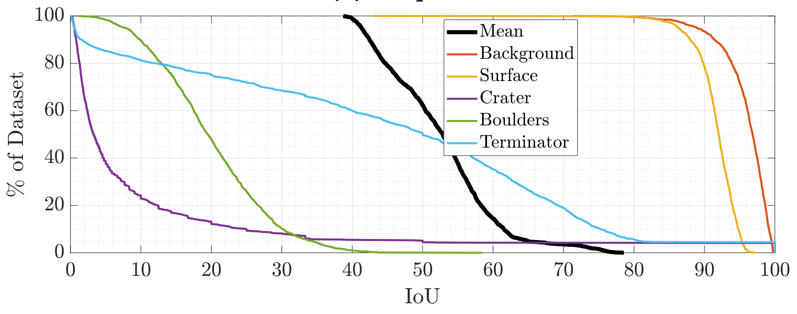
A mosaic is illustrated in Figure 4.7 with input images, true, and predicted masks for $\mathcal{DS}_1^{D-1}$.

In Figure 4.8 it is possible to see the network performance in a flyby scenario, which is an interesting application to enhance its scientific return [54]. It is possible to appreciate the increased accuracy in the detection of boulders as the camera gets closer to the body, as well as the overall difficulty in robustly detecting craters.

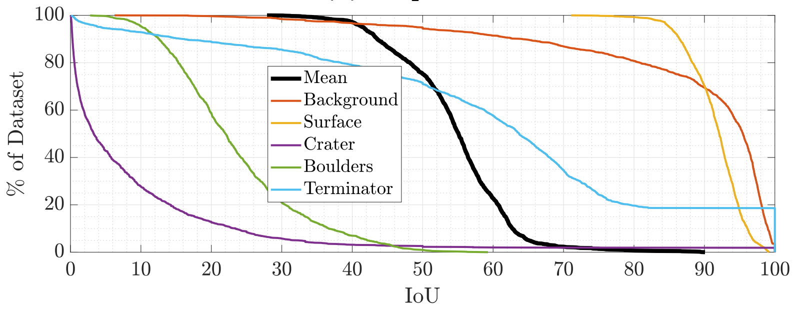
It is also commented that the shape model of Thisbe used in $\mathcal{DS}_1^{D-2b}$ and $\mathcal{DS}_1^{D-3}$ is designed to challenge the network by positioning two nested craters over the viewing face of the body during the flyby. This feature has never been seen during training and has been introduced to assess the network generalization capabilities. As it is possible to see from the mosaic in Figure 4.9, the double crater is robustly identified only in a few cases, while single craters are identified more robustly in the same dataset. Finally, it is also commented that network predictions



(a) $\mathcal{DS}_1^{D-1}$.



(b) $\mathcal{DS}_1^{D-2a}$.



(c) $\mathcal{DS}_1^{D-2b}$.

Figure 4.6: Cumulative mIoU and IoU (coloured) for each class of the UNet_S on the test set of $\mathcal{DS}_1^{D-1}$ (a), $\mathcal{DS}_1^{D-2a}$ (b), and $\mathcal{DS}_1^{D-2b}$ (c).

are unstable from considerable distances from the target body, while they become significantly more robust closer to the surface. This behavior is expected when features occupy only a few pixels in the input image.

4.1.3.2 Segmentation over $\mathcal{DS}_1^{D-4}$

The performance of UNet_S, $\text{UNet}_\text{S}^\text{A}$, and UNet_R over the dataset made of real images, $\mathcal{DS}_1^{D-4}$, are summarized in Table 4.6.

As expected, the worst-performing network is UNet_S, since it has been trained only over synthetic images. The drop in performance hints at the existence of a domain gap between synthetic and real images. This seems particularly relevant

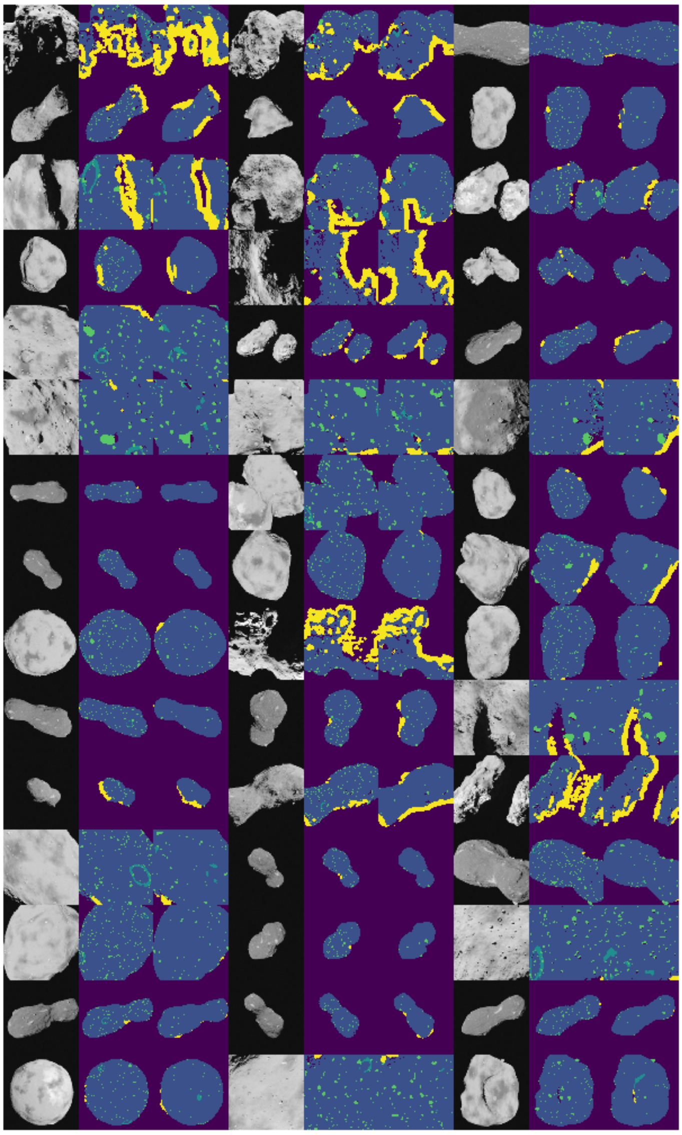


Figure 4.7: Mosaic of *15* $\times$ *3* triplets showing input (left), true mask (center) and predicted mask (right) for test dataset of $\mathcal{DS}_1^{D-1}$ by UNet_S. See Figure 4.2 for the legend of the layers.

for craters and boulders and only marginally for the terminator region.

Performance is recovered with $\mathrm{UNet}_\mathrm{S}^\mathrm{A}$ and UNet_R, with the first performing slightly better than the second, as it is possible to see from the mosaics in Figure 4.10.

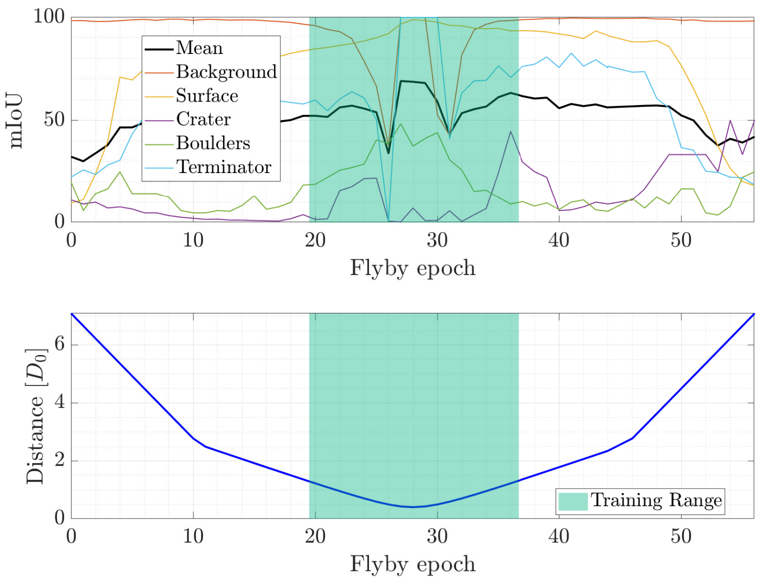


Figure 4.8: Cumulative mIoU (black) and IoU (coloured) for each class of the UNet_S in test set of $\mathcal{DS}_1^{D-3}$ as function of time (top). Distance from the asteroid as a function of time (bottom).

From the former, it is possible to appreciate the poor performance of UNet_S in detecting craters, hinting at the unrealistic crater modeling in the synthetic images of $\mathcal{DS}_1$.

The poor performance of UNet_S with real images justifies the existence of $\text{UNet}_\text{S}^\text{A}$ and UNet_R. Indeed, these two networks are designed for operative scenarios: use the few real images that could be downlinked to the ground once a spacecraft arrives at the target body to train them. Indeed, $\mathcal{DS}_1^{D-4}$ is made only by as little as *50* images (from 4 different bodies) that, using data augmentation, are virtually increased to *200* images. This also reflects the original methodology in [136], which used data augmentation on a severely limited set of only *30* images available for training for a biomedical application.

The remaining question remains whether or not to perform bootstrap learning by using a pre-trained network with synthetic images ($\text{UNet}_\text{S}^\text{A}$) or to solely rely on limited real images for (UNet_R). Looking at the performance achieved by the two networks, the bootstrap strategy delivers better results.

Figure 4.11 displays the mIoU of the validation curve during training of the three best final instances of the architectures $\text{UNet}_\text{S}^\text{A}$ and UNet_R. From this figure, it is possible to appreciate a faster and smoother convergence by $\text{UNet}_\text{S}^\text{A}$ to a higher performance plateau than the one reached by UNet_R. It is also observed that since the early phases of the training, $\text{UNet}_\text{S}^\text{A}$ is advantaged by the training experience of UNet_S, starting from higher values of mIoU. These same values are reached by UNet_R only after *150 – 200* epochs.

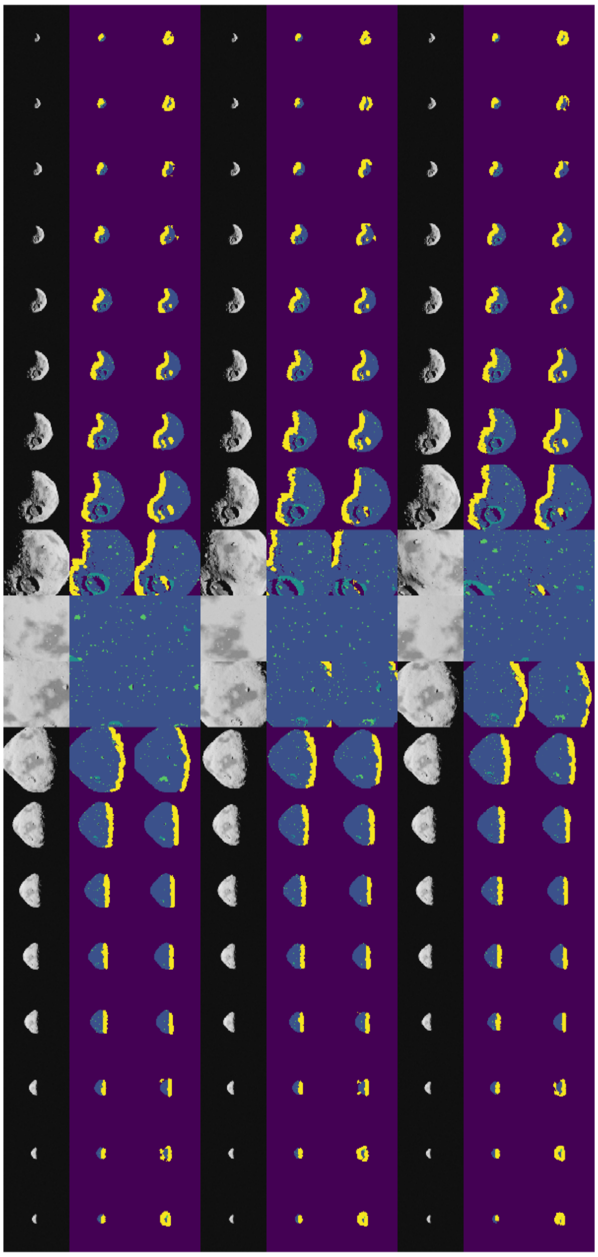
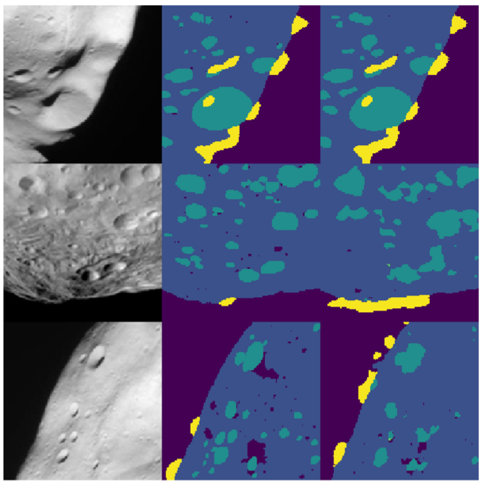


Figure 4.9: Mosaic of 19×3 triplets showing input (left), true mask (center), and predicted mask (right) for the test set of $\mathcal{DS}_1^{D-3}$ by UNet$_S$. See Figure 4.2 for the legend of the layers.

Table 4.6: Summary of the mIoU for $\mathcal{DS}_1^{D-4}$ for the different test cases expressed as a percentage.

Test case		UNet$_\mathrm{S}$	UNet$_\mathrm{S}^\mathrm{A}$	UNet$_\mathrm{R}$
	min	1.16	1.14	15.95
Background	mean	**69.84**	**76.79**	**74.00**
	max	92.19	98.53	98.28
	min	59.18	73.90	67.71
Surface	mean	**82.00**	**88.33**	**83.58**
	max	93.00	96.84	96.21
	min	0.04	1.06	0.30
Craters	mean	**12.63**	**56.94**	**49.47**
	max	100.00	100.00	100.00
	min	0.09	2.78	1.65
Boulders	mean	**4.59**	**67.42**	**65.31**
	max	20.00	100.00	100.00
	min	1.28	3.33	6.04
Terminator	mean	**24.64**	**55.68**	**52.68**
	max	100.00	100.00	100.00
Mean		**38.74**	**69.03**	**65.01**


(a) Predictions by UNet$_\mathrm{S}^\mathrm{A}$.

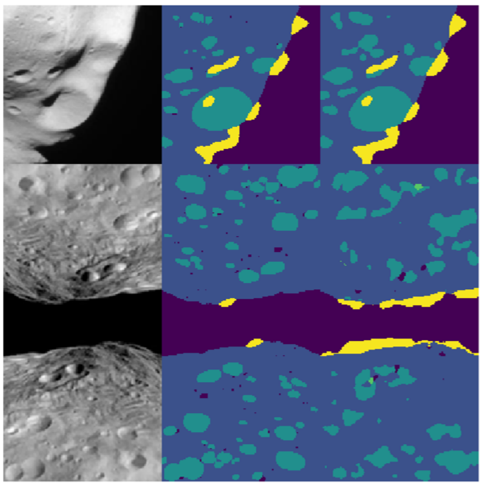

(b) Predictions by UNet$_\mathrm{R}$.

Figure 4.10: input (left), ground truth (center), and predicted (right) masks in the best (top), average (middle), and worst (bottom) cases of the test set of $\mathcal{DS}_1^{D-4}$ by UNet$_\mathrm{S}^\mathrm{A}$ (a) and UNet$_\mathrm{R}$ (b). See Figure 4.2 for the legend of the layers.

These results justify the preference for the bootstrap training strategy, the importance of defining a synthetic dataset, and how a small, manually labeled dataset of real images can tune a network to perform in an operational scenario. The predictions of UNet$_\mathrm{S}$, UNet$_\mathrm{S}^\mathrm{A}$, and UNet$_\mathrm{R}$ are visualized one next to each other

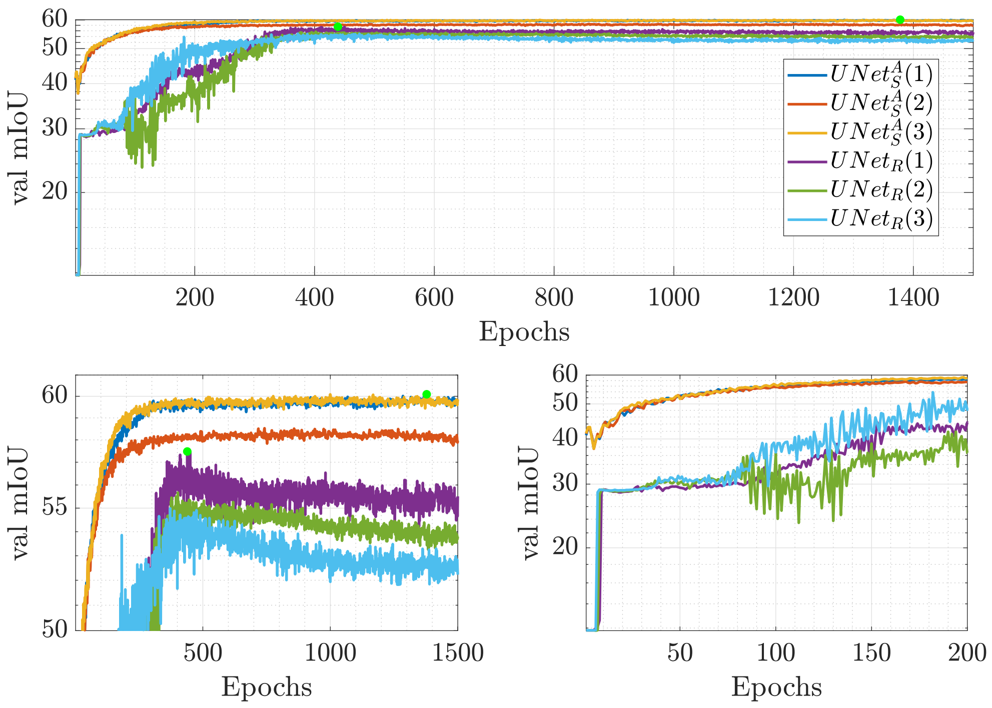


Figure 4.11: Validation mIoU for the three best training of $\text{UNet}_\text{S}^\text{A}$ and UNet_R. (Top) global view, (bottom-left) zoom on the higher values of mIoU, (bottom-right) zoom early on in the training.

with test images of $\mathcal{DS}_1^{D-4}$ in Figure 4.12.

4.1.3.3 Segmentation with hybrid architecture

Applying UNet_H to $\mathcal{DS}_1^{D-4}$, the segmentation maps illustrated in Figure 4.13 are generated, from which a qualitative assessment of UNet_H can be performed.

As commented before, large boulders (as the ones manually labeled on $\mathcal{DS}_1^{D-4}$) are better predicted with lower values of γ, and vice-versa for small boulders (as the ones automatically labeled on $\mathcal{DS}_1^{D-1}$). This is intended as γ weights the contribution of UNet_S and $\text{UNet}_\text{S}^\text{A}$ in predicting this layer.

The approach implemented with UNet_H benefits from the simplicity of training with only synthetic images and from dedicated fine-tuning performed on selected features such as craters and boulders. This could be efficient in an operational scenario by deploying an architecture trained with synthetic images of the assumed surface before arrival and progressively fine-tuning it with data acquired in the first phases of the mission. By relying on a well-characterized architecture with synthetic images, mission designers can always weigh its contribution dynamically by varying γ depending on the confidence in the different architectures for the specific operational range considered. Ultimately, γ can prioritize various boulders' sizes depending on the scenario.

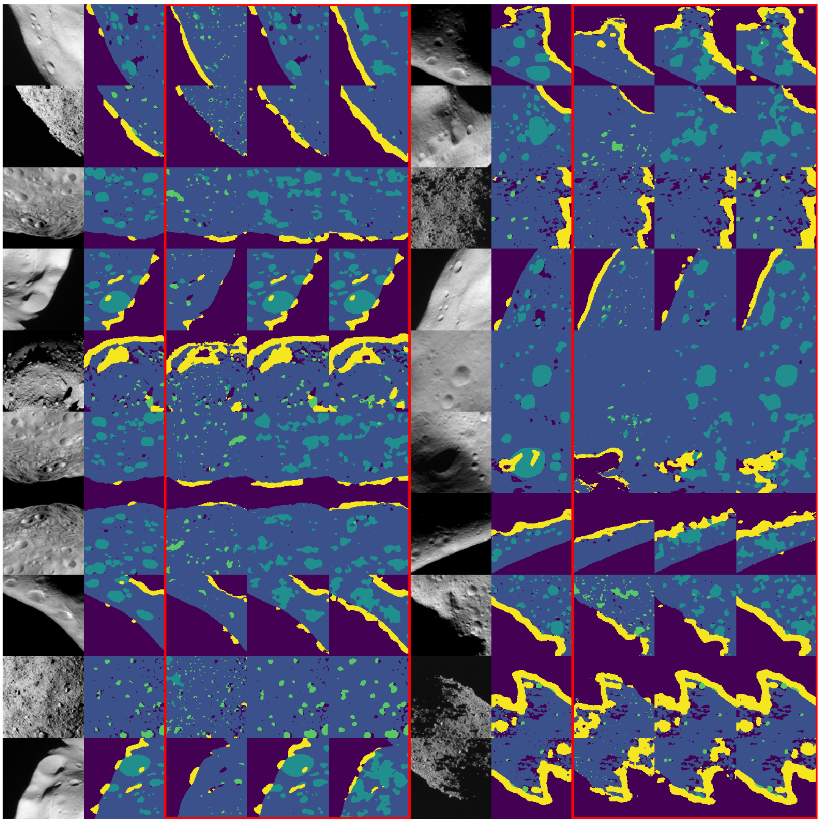
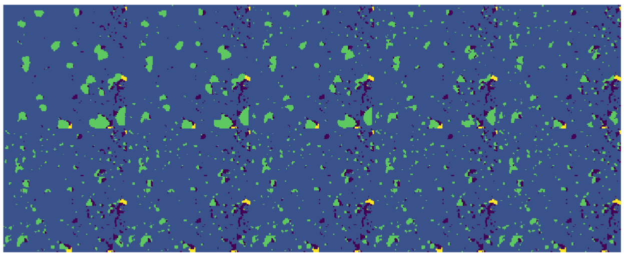


Figure 4.12: Mosaic of 10×2 quintuplets showing input image(left), ground truth (center) and predicted mask (from left to right) of UNet_S, $\text{UNet}_\text{S}^\text{A}$, and UNet_R. The latter three predictions are highlighted in red. See Figure 4.2 for the legend of the layers.

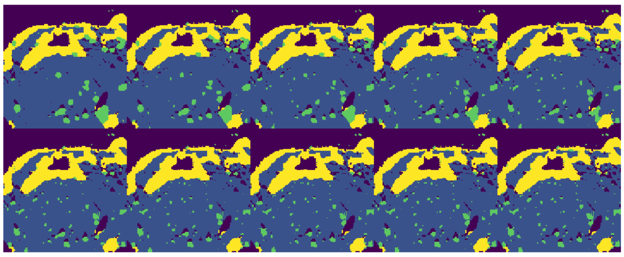
4.1.3.4 Uncertainty quantificaton

Finally, the uncertainty maps created with the methodology presented in Section 4.1.2 are illustrated in this section using the test set of $\mathcal{DS}_1^{D-4}$. First, the usage of the uncertainty maps to substitute semantic pixels is illustrated in Figure 4.14 on predictions from $\text{UNet}_\text{S}^\text{A}$. The red pixels represent those in the 6^{th} layer, corresponding to the uncertain class.

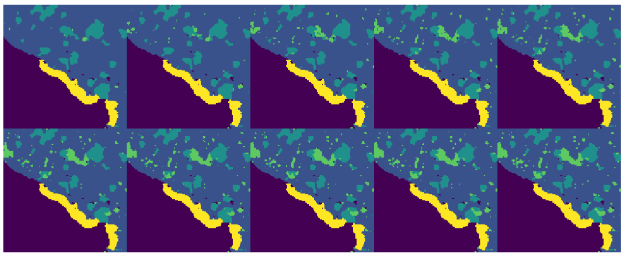
The masks enhanced by the $6-th$ layer contain uncertain pixels only at the boundaries between different classes. Since pixel classification is only being challenged on the borders, the operational inclusion of uncertainty does not bring relevant improvement. When the mIoU is evaluated on the enhanced maps, the network's overall performance improves only marginally. The maximum change is represented by a positive 1.4% variation on mIoU. Finally, it is also commented that a test case where the second most probable class substitutes the uncertain



(a) UNet_H on sample 1.



(b) UNet_H on sample 2.



(c) UNet_H on sample 3.

Figure 4.13: Example of hybrid predictions from UNet_H with different weights. From left to right, top to bottom γ changes from 1 to 10. See Figure 4.2 for the legend of the layers.

pixels has also been investigated but provided limited gains in performance. A larger sample of the uncertainty maps generated by UNet_S, $\text{UNet}_\text{S}^\text{A}$, and UNet_R is illustrated in Figure 4.15, showcasing that the networks generate similar uncertainty predictions.

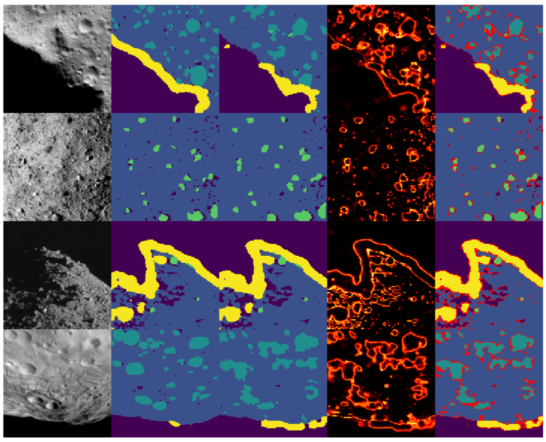


Figure 4.14: Mosaic view of $\text{UNet}_\text{S}^\text{A}$ input (1st column), true and predicted masks (2nd and 3rd), uncertainty map (4th) and enhanced predicted mask (5th) for four samples of $\mathcal{DS}_1^{D-4}$. Red pixels belong to the 6^{th} layer of highly uncertain pixels. See Figure 4.2 for the legend of the layers.

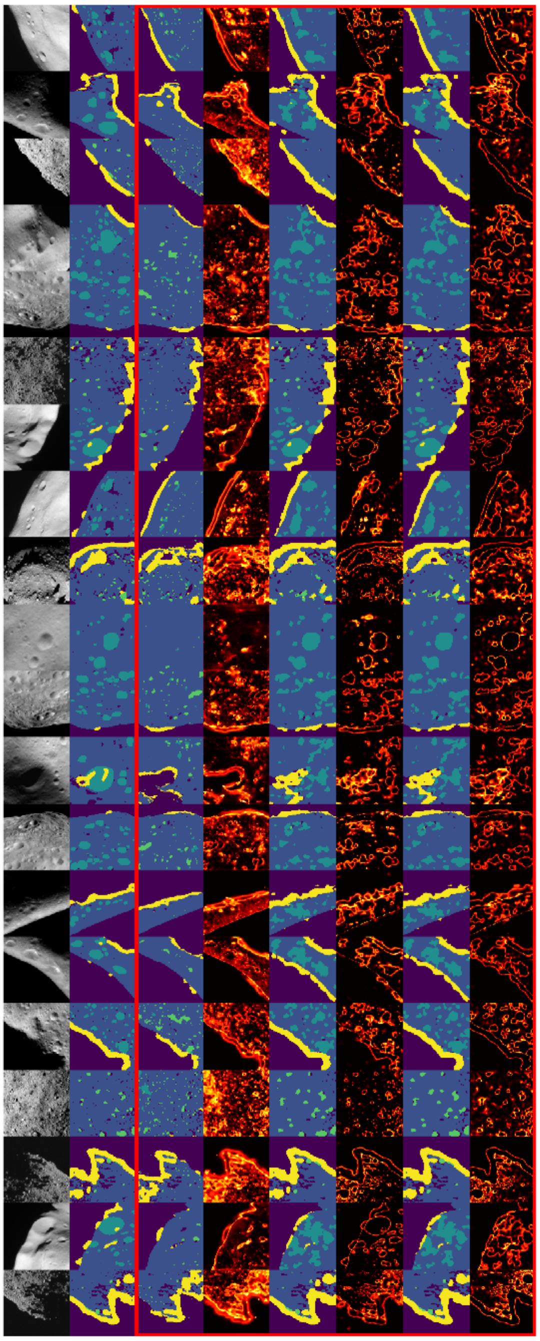


Figure 4.15: Mosaic of 20 samples from the test set of $\mathcal{DS}_1^{D-4}$ showing input (1st column), true mask (2nd column) together with predicted mask and uncertainty map pairs of UNet_S, $\text{UNet}_\text{S}^\text{A}$, and UNet_R (from left to right in the red rectangle). See Figure 4.2 for the legend of the layers.

4.2 Boulders segmentation

The robust identification of boulders on the surface of small bodies can impact both features-based navigation methods and hazard detection and avoidance algorithms for landing applications. Moreover, as highlighted in the previous section, previously designed pipelines struggle to segment boulders at different scales, requiring hybrid predictions.

To address these limitations, this section focuses on the robust image segmentation of boulders at different scales and under varying illumination conditions.

4.2.1 Training strategy

An incremental training strategy of *4* steps involving different architectures and training paradigms is designed to develop a robust image segmentation network. The strategy is intended to efficiently accompany the design of the final architecture by incrementally training some of its portions using different datasets. The dataset used for this activity is described in detail in Section A.1.2 and is referred to as $\mathcal{DS}_2$. $\mathcal{DS}_2$ is made of three datasets: $\mathcal{DS}_2^{DS1}$, $\mathcal{DS}_2^{DS2}$, and $\mathcal{DS}_2^{DS3}$. $\mathcal{DS}_2^{DS1}$ is made by image-label pairs of single procedural boulders imaged over a spherical surface and is used for regression and segmentation. $\mathcal{DS}_2^{DS2}$ is made by image-label pairs of multiple boulders scattered across the shape model of Didymos [138]. Lastly, $\mathcal{DS}_2^{DS3}$ is made by image-label pairs of real images in which boulders are manually labeled. This is the same set used in the previous section as part of $\mathcal{DS}_1^{D-4}$.

The overall training process is schematized in Figure 4.16.

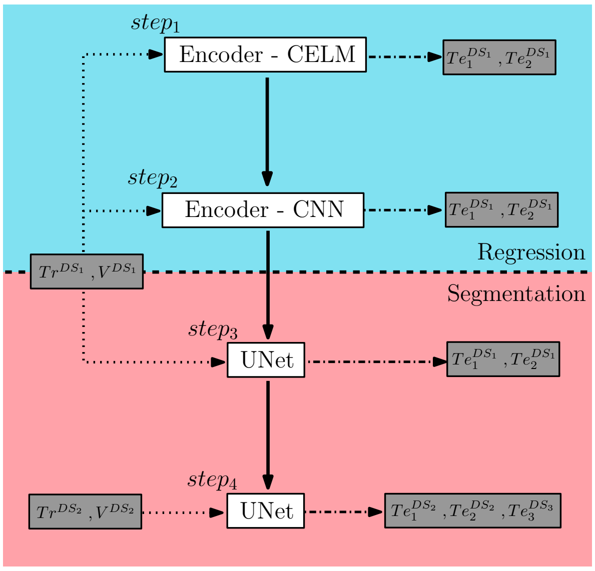


Figure 4.16: Schematic of the training procedure adopted in this section.

CELM theory is used in *step*$_1$ to expedite the architecture design search of an encoder, which is further refined in *step*$_2$ as a conventional CNN. Similarly to the strategy presented in the previous section, the encoder is trained over a proxy regression task on the $\mathcal{DS}_2^{DS1}$ dataset to predict the Center of Brightness (CoB) of single boulders appearing in the images. The CELM framework is exploited as an effective tool to efficiently explore the architecture design space. Figure 4.17 showcases an example of how to use global metrics to discern between different architectures (out of the *1134* combinations considered) as a function of the performance in combination with the hyperparameters chosen.

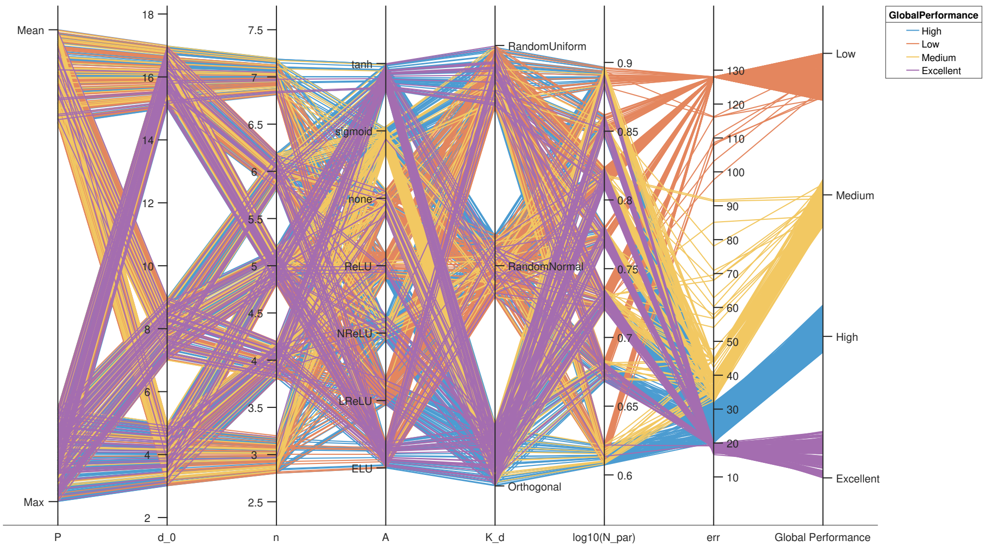


Figure 4.17: Parallel-plot showing the dependencies between the different hyperparameters illustrated in Table 4.7 and the network performances. The lines are colored by four quality metrics related to the performance: excellent, high, medium, and low.

Once a first exploration is performed, the best-performing architecture of the pool of the ones tested is chosen for implementation and is generally referred to as the CELM-encoder. The total time needed to train the *1134* architectures with the CELM paradigm and the hardware available is equivalent to *48.3* hours. On average, *13.93%* of the time is spent on the forward pass of the validation tensor, *81.32%* on the forward pass of the training tensor, while the remaining *4.75%* is spent solving the least square equation (See Eq. (2.25)). The time saved exploring the architecture design space with CELM, irrespective of the hardware used during training, can be crucial to fast forward and ease the overall training.

In *step*$_2$, the weights and biases of the single best-performing encoder are optimized via a standard CNN training using MBGD. During this step, the architectural elements are frozen while the weights and biases of the kernels are optimized, varying the batch sizes (64, 128, 256, 512) and learning rates (10^{-4}, 10^{-3}, 10^{-2}). Each setup is initialized and runs randomly two times for short epochs for *24* training episodes. The best-performing ones are selected based on the error achieved over the validation split during the entire training and are re-run while increasing the epochs. The loss function used to train the network is the MSE.

The final setup is achieved using a batch size $\mathcal{B}$ of *32* samples, with a learning rate *lr* of 10^{-4}, and a dropout rate in the fully connected layer of *0.2*.

Note that the training time of the best-performing architecture for *200* epochs using the MBGD method required a total of *9504.7* s, while the equivalent training time with a CELM would have taken on average *153* s per architecture (spent primarily on the forward pass to generate **H** for the training and validation sets). The total training time to find the best learning rate and batch size combination in $step_2$ is roughly *24* h. Should all the *1134* architectures have been trained using the MBGD, considering the encoder as a CNN, the exploration of the architecture design space would have resulted in a much more computationally expensive process.

A partial training of the UNet for segmentation is then performed in $step_3$ using $\mathcal{DS}_2^{DS1}$, which is further refined in $step_4$ with the use of the $\mathcal{DS}_2^{DS2}$ dataset.

$step_3$ exploits the CNN-encoder refined from the previous step and inserts it into a larger architecture designed for segmentation. A UNet [136] architecture is considered for such a task. Similarly to [134], the UNet is trained by incrementally increasing the epochs while testing various combinations of dropout values, batch size, learning rate, and depth of the decoder layers. A WSCCE is used as a loss function, while the mIoU is used as a metric. The weights for the WSCCE loss are computed from statistical analysis of the pixel content in the masks of the training set of $\mathcal{DS}_2^{DS1}$. As *3.99%* of the pixels are boulders while *93.01%* are not, the complement of these values are used respectively as weights of the non-boulder and boulder classes. The best-performing architecture has been found with a dropout equal to *0.2*, a batch of *256*, a learning rate of *0.001*, and decoder depths of *192*,*96*,*48*,and *24*. The total training time spent to find out this setup equals *18.9* hours.

Finally, in the fourth and last step of the training, the same procedure illustrated in the previous step is repeated using the $\mathcal{DS}_2^{DS2}$ dataset. In this case, the training of the UNet is not initialized from scratch but starts from the set of weights and biases found in the previous step to help the network plateau at higher values of the mIoU on the validation set. This allows the network to achieve better performance and generalization capabilities than it would by starting from scratch, as exemplified by the results in the previous section. The WSCCE loss function uses weights corresponding to *27.52%* and *72.48%*, respectively, for the non-boulder and boulder classes. The final architecture is trained over *400* epochs with a learning rate of *0.001* and a batch size of *16*. The total training time spent to find out this setup is equal to *12* hours.

This incremental approach ultimately allows better generalization and improved performance of the final UNet compared to a training executed from scratch considering only $step_4$. The entire training to obtain the final IP network took roughly *103* hours[30] from start to finish.

[30]Using a Tesla P100-PCIE 16Gb GPU, with a 27.3 Gb of RAM in Google colab

4.2.2 Convolutional architectures

The encoder is designed as a sequence of cells C_i operating tensor batches, each composed of a combination of dilated convolutions, activation functions, and pooling layers, as exemplified in Figure 4.18. In these architectures, dilated convolutions are investigated for their beneficial effects in augmenting the receptive field of the kernels as well as their observed capability to boost segmentation performance [22, 139, 140]. Dilated convolution with rates 1, 2, and 3 are investigated in the architectures. The outputs of the convolutions at different dilation rates are stacked together as illustrated in Figure 4.18.

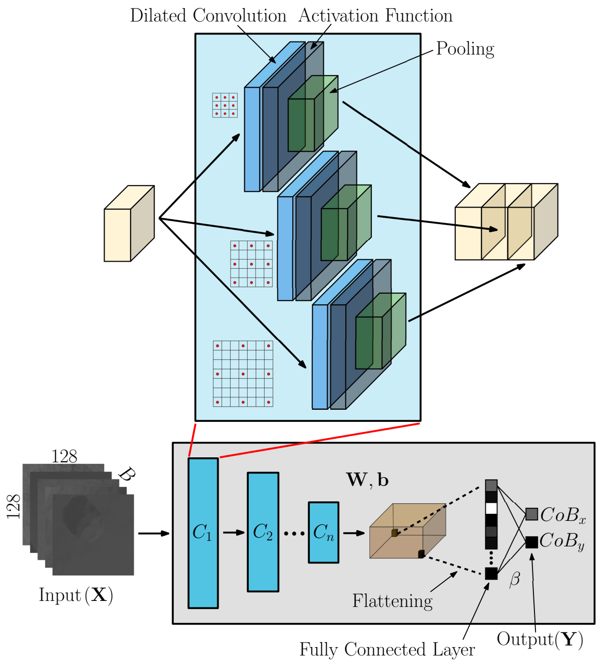


Figure 4.18: Schematic of the architecture of the encoder with dilated convolutions.

The architecture with the best capacity is found by defining a set of rules to build up the encoder. Assuming a constant kernel size of 3×3 and an exponential depth expansion coefficient equal to 2, different architectures are generated by varying the pooling strategy P, the initial depth of the network d_0, the total number of cells n, the activation functions A, the kernel initialization strategy K_d, and the number of random runs for each architecture N_r with the combinations illustrated in Table 4.7. The regularization parameter C is varied for each architecture as 10^{-3},10^{-2},10^{-1},1,10^{1},10^{2},10^{3}. By combining these parameters, a total of 1134 different architectures are generated.

The setup achieving the best performance in predicting boulder's CoB in the images is chosen as the champion architecture to be considered in the next steps of the training and is represented in bold in Table 4.7.

Both CELM and CNN share an identical architecture, the only difference being

Table 4.7: Summary of the hyperparameters used in this section to search for the optimal architecture of the encoder (represented in bold).

Name	Values
P	mean, **max**
d_0	*4*, *5*, **16**
n	$(3,4,5)_{d_0=4}$, $(4,\mathbf{5},6)_{d_0=8}$, $(5,6,7)_{d_0=16}$
A	NReLU, ReLU, LReLU, **ELU**, tanh, sigmoid, none
K_d	RandomUniform (-1,1), RandomNormal (0,1), **Orthogonal**
N_r	**3**

the training strategy adopted. The encoder architecture is summarized in Table 4.8.

On the other hand, the segmentation architecture is illustrated in Table 4.9.

4.2.3 Results

This section illustrates the performance of the encoder and segmentation architectures introduced in the previous section on the test sets of $\mathcal{DS}_2$.

4.2.3.1 Performance of the encoder

The performances of the two encoders are illustrated in Table 4.10 for the two test sets of $\mathcal{DS}_2^{DS1}$ using as metric the coordinates of the error in the CoB estimation as $\varepsilon_{\mathbf{CoB}} = \mathbf{CoB}^e - \mathbf{CoB}^t$.

As expected, the error of the CNN is lower than the one of the CELM, albeit the difference is small. It is also observed that both encoders exhibit limited variability between the two test sets of $\mathcal{DS}_2$.

For curiosity, the histograms of the distributions of β_u, β_v, $\mathbf{W}_u$ and $\mathbf{W}_v$ (only between the last hidden layer and the output layer), are illustrated in Figure 4.19(a) and Figure 4.19(b).

Both sets of weights are normally distributed. However, it is noted that those of the CNN exhibit a variance one order of magnitude smaller. It is not clear if this difference can directly influence the performance or if better generalization capabilities of the CELM could have been reached with smaller values of β (in theory, lower values of the regularization term C had been used on the validation set during training of the CELM that should have lowered β, but they did not cause better performance on the validation set).

4.2.3.2 Performance of the UNet

The performances of the best UNet architecture generated after *step*$_3$ and *step*$_4$ are summarized in Table 4.11 respectively on the test sets of $\mathcal{DS}_2^{DS1}$, $\mathcal{DS}_2^{DS2}$, and $\mathcal{DS}_2^{DS3}$.

Table 4.8: Detailed architecture of the encoder, made of *3′527′040* parameters (13.45 MB), all of which are trainable.

ID	Layer type	Output Shape	Parameters	Connections
I	InputLayer	($\mathcal{B}$, 128, 128, 1)	0	C11,C12,C13
C11	Conv2D	($\mathcal{B}$, 128, 128, 16)	160	CC1
C12	Conv2D	($\mathcal{B}$, 128, 128, 16)	160	CC1
C13	Conv2D	($\mathcal{B}$, 128, 128, 16)	160	CC1
CC1	Concatenate	($\mathcal{B}$, 128, 128, 48)	0	A1
A1	Activation	($\mathcal{B}$, 128, 128, 48)	0	P1
P1	Pooling	($\mathcal{B}$, 64, 64, 48)	0	C21,C22,C23
C21	Conv2D	($\mathcal{B}$, 64, 64, 32)	13856	CC2
C22	Conv2D	($\mathcal{B}$, 64, 64, 32)	13856	CC2
C23	Conv2D	($\mathcal{B}$, 64, 64, 32)	13856	CC2
CC2	Concatenate	($\mathcal{B}$, 64, 64, 96)	0	A2
A2	Activation	($\mathcal{B}$, 64, 64, 96)	0	P2
P2	Pooling	($\mathcal{B}$, 32, 32, 96)	0	C31,C32,C33
C31	Conv2D	($\mathcal{B}$, 32, 32, 64)	55360	CC3
C32	Conv2D	($\mathcal{B}$, 32, 32, 64)	55360	CC3
C33	Conv2D	($\mathcal{B}$, 32, 32, 64)	55360	CC3
CC3	Concatenate	($\mathcal{B}$, 32, 32, 192)	0	A3
A3	Activation	($\mathcal{B}$, 32, 32, 192)	0	P3
P3	Pooling	($\mathcal{B}$, 16, 16, 192)	0	C41,C42,C43
C41	Conv2D	($\mathcal{B}$, 16, 16, 128)	221312	CC4
C42	Conv2D	($\mathcal{B}$, 16, 16, 128)	221312	CC4
C43	Conv2D	($\mathcal{B}$, 16, 16, 128)	221312	CC4
CC4	Concatenate	($\mathcal{B}$, 16, 16, 384)	0	A4
A4	Activation	($\mathcal{B}$, 16, 16, 384)	0	P4
P4	Pooling	($\mathcal{B}$, 8, 8, 384)	0	C51,C52,C53
C51	Conv2D	($\mathcal{B}$, 8, 8, 256)	884992	CC5
C52	Conv2D	($\mathcal{B}$, 8, 8, 256)	884992	CC5
C53	Conv2D	($\mathcal{B}$, 8, 8, 256)	884992	CC5
CC5	Concatenate	($\mathcal{B}$, 8, 8, 768)	0	A5
A5	Activation	($\mathcal{B}$, 8, 8, 768)	0	P5
P5	Pooling	($\mathcal{B}$, 4, 4, 768)	0	FC
FC	Flatten	($\mathcal{B}$, 12288)	0	DO
DO	Dropout	($\mathcal{B}$, 12288)	0	O
O	Dense	($\mathcal{B}$, 2)	24578	

As for the encoder, the network performs similarly in Te_1 and Te_2, indicating that the two test sets do not offer different conditions for the network's capability. While the mIoU is very high for the test sets in $\mathcal{DS}_2^{DS1}$, the same is not valid for the ones in $\mathcal{DS}_2^{DS2}$ and $\mathcal{DS}_2^{DS3}$. Indeed, a large population of multiple boulders seems to challenge the capabilities of the UNet, which is lower than the one trained with single boulders in $step_3$. This is also substantiated by the WSCCE (which is two orders of magnitude higher than in the previous case, indicating a difficulty encountered during training that did not specialize the network for high performance when segmenting multiple boulders). Similar trends are observed for the other metrics, A and $MIOU$.

Nonetheless, the performance achieved by such a network is an improvement of the ones presented in Section 4.1. Specializing only for boulder segmentation

Table 4.9: Detailed architecture of the UNet, made of 5'510'066 parameters (21.02 MB), 1'982'306 of which are trainable (7.56 MB).

ID	Layer type	Output Shape	Parameters	Connections
I	InputLayer	($\mathcal{B}$, 128, 128, 1)	0	E1
E1	Encoder	($\mathcal{B}$, 128, 128, 48)	480	E2, CC4
E2	Encoder	($\mathcal{B}$, 64, 64, 96)	41568	E3, CC3
E3	Encoder	($\mathcal{B}$, 32, 32, 192)	166080	E4, CC2
E4	Encoder	($\mathcal{B}$, 16, 16, 384)	663936	E5, CC1
E5	Encoder	($\mathcal{B}$, 8, 8, 768)	2654976	CT1
CT1	Sequential	($\mathcal{B}$, 16, 16, 192)	1327872	CC1
CC1	Concatenate	($\mathcal{B}$, 16, 16, 576)	0	DO1
DO1	Dropout	($\mathcal{B}$, 16, 16, 576)	0	A1
A1	Activation	($\mathcal{B}$, 16, 16, 576)	0	CT2
CT2	Sequential	($\mathcal{B}$, 32, 32, 96)	498048	CC2
CC2	Concatenate	($\mathcal{B}$, 32, 32, 288)	0	DO2
DO2	Dropout	($\mathcal{B}$, 32, 32, 288)	0	A2
A2	Activation	($\mathcal{B}$, 32, 32, 288)	0	CT3
CT3	Sequential	($\mathcal{B}$, 64, 64, 48)	124608	CC3
CC3	Concatenate	($\mathcal{B}$, 64, 64, 144)	0	DO3
DO3	Dropout	($\mathcal{B}$, 64, 64, 144)	0	A3
A3	Activation	($\mathcal{B}$, 32, 32, 144)	0	CT4
CT4	Sequential	($\mathcal{B}$, 128, 128, 24)	31200	CC4
CC4	Concatenate	($\mathcal{B}$, 128, 128, 72)	0	DO4
DO4	Dropout	($\mathcal{B}$, 128, 128, 72)	0	A4
A4	Activation	($\mathcal{B}$, 32, 32, 72)	0	O
O	Conv2DTranspose	($\mathcal{B}$, 128, 128, 2)	1298	

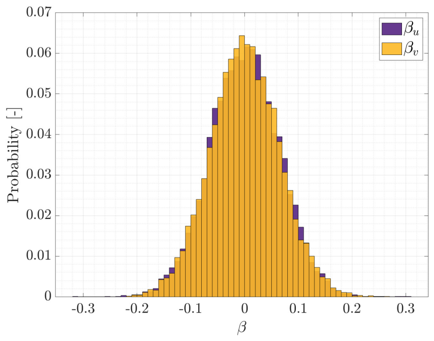
Table 4.10: Performance of the encoders on the test sets of $\mathcal{DS}_2^{DS1}$.

Encoder	Dataset	$\mu\varepsilon^u_{CoB}$ [px]	$\mu\varepsilon^v_{CoB}$ [px]
CELM	Te1	16.36	11.46
CELM	Te2	16.30	11.36
CNN	Te1	7.01	7.30
CNN	Te2	7.06	7.40

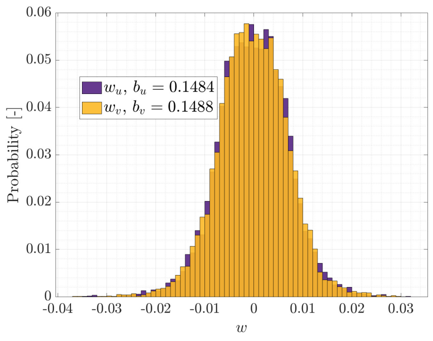
made it possible to develop a better network. When a single boulder is imaged over the surface, the network is observed to detect its presence robustly under various illumination conditions.

Figure 4.20, Figure 4.21, and Figure 4.22 showcase random samples of input images, true and predicted masks of test images from $\mathcal{DS}_2^{DS1}$, $\mathcal{DS}_2^{DS2}$, and $\mathcal{DS}_2^{DS3}$, respectively.

The very high performances of the network when considering isolated boulders are reflected in the well-predicted masks in Figure 4.20. Such capability is also partially transferred to the subsequent architecture trained over multiple boulders scattered across the surface. As it is possible to see in Figure 4.21, the network can correctly predict many boulders with varying geometric and illumination conditions.



(a) CELM encoder.



(b) CNN encoder.

Figure 4.19: Distribution of β (a) and **w** (b) in the last layer of the CELM-encoder (a) and CNN-encoder (b). The y-axis shows the relative probability of each bin using a bindwidth of *0.01* (a) and *0.001*.

Table 4.11: UNet performance on the test sets of $\mathcal{DS}_2$.

Dataset	$\mu(WSCCE)$ $[-]$	$\mu(A)$ $[\%]$	$\mu(MIOU)$ $[\%]$
$\mathcal{DS}_2^{DS1}$ Te1	7.4 10^{-3}	99.19	90.78
$\mathcal{DS}_2^{DS1}$ Te2	9.6 10^{-3}	99.20	91.03
$\mathcal{DS}_2^{DS2}$ Te1	1.59 10^{-1}	82.22	66.09
$\mathcal{DS}_2^{DS2}$ Te2	1.6 10^{-1}	81.98	66.04
$\mathcal{DS}_2^{DS3}$ Te3	2.8 10^{-1}	62.53	33.26

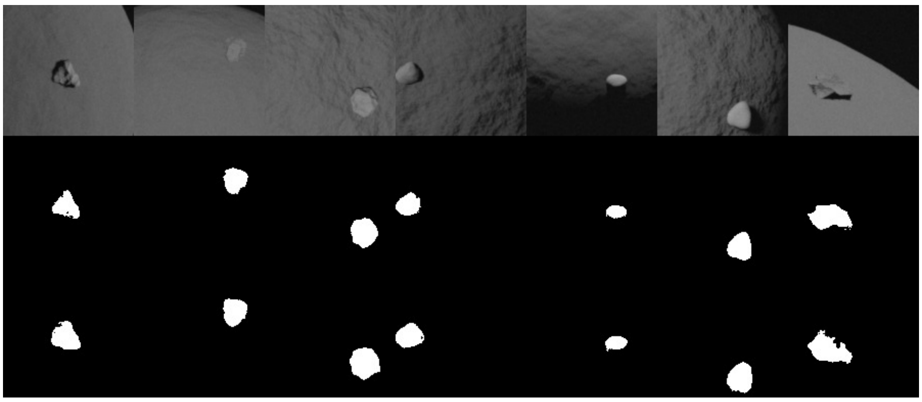


Figure 4.20: Samples of the input image (top), true mask (middle), and predicted mask (bottom) from the UNet trained in *step*$_3$ on test images of $\mathcal{DS}_2^{DS1}$.

It is also noted that the true masks exhibit challenging conditions in which a dense boulder's presence makes the surface almost entirely covered. This reflects actual environmental conditions, such as the ones encountered in Ryugu and Bennu.

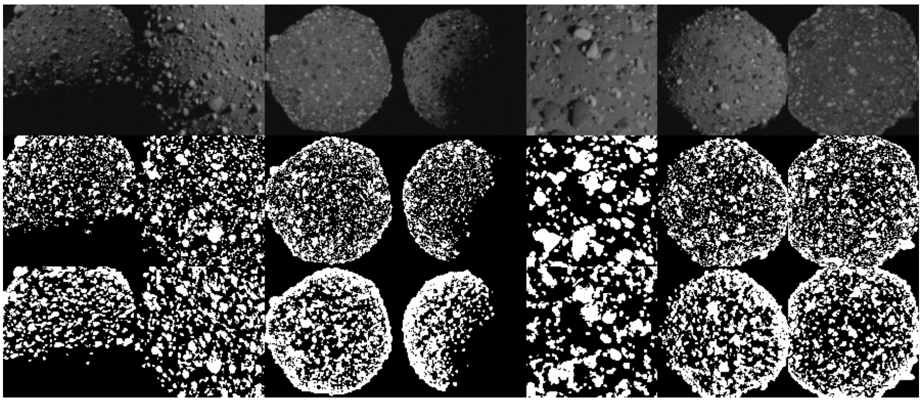


Figure 4.21: Samples of the input image (left), true mask (center), and predicted mask (right) from the UNet trained in *step$_4$* on test images of $\mathcal{DS}_2^{DS2}$.

From the predicted masks in Figure 4.21, it is also possible to observe an incorrect network behavior over the edge of the body. The network incorrectly predicts boulders' presence in this region, driving down its performance. This seems to happen as the ray-tracing algorithm in Blender correctly labels true boulders over the projection of the edge in the image plane even when only a few pixels are observed over the very edge of the body, as it is also possible to observe from the true masks in Figure 4.21. Such a labeling mishap may have promoted the network to classify the entire edge of the target body as a boulder, which is an unwanted behavior. The same phenomenon is not observed over the terminator region of the body since the shadows nullify the boulder's label in the mask (the mask of the boulder is obtained as a multiplication between the boulder's object and the illumination condition, so shadowed portions of a boulder are not labeled in the true masks used for training).

Lastly, the final UNet is tested to predict boulders on real images from previously flown missions. First of all, as it is possible to see in Figure 4.22, it is commented that the noise levels from real sensors do not pose particular challenges to the network in terms of generalization. Albeit the network has been trained without a tailored noise setup modeling any specific camera, the network performs similarly in all types of images considered. Also, note that images from $\mathcal{DS}_2^{DS3}$ are captured with different sensors and illumination conditions, all factors that are not impacting the performance.

Is it thought that thanks to the injection of artificial noise as well as thanks to the particular care that has been put into the design of domain randomization strategies in the generation of $\mathcal{DS}_2^{DS1}$ and $\mathcal{DS}_2^{DS2}$, the network developed generalization across a variety of changes in the input space. The only case in which this has been observed to generate minor artifacts is visualized in the mosaic in Figure 4.22, in the second column, third-row sample.

Finally, it is remarked that the current masks in $\mathcal{DS}_2^{DS3}$ were inappropriate for this specific design. These are the same masks used in Section 4.1, manually

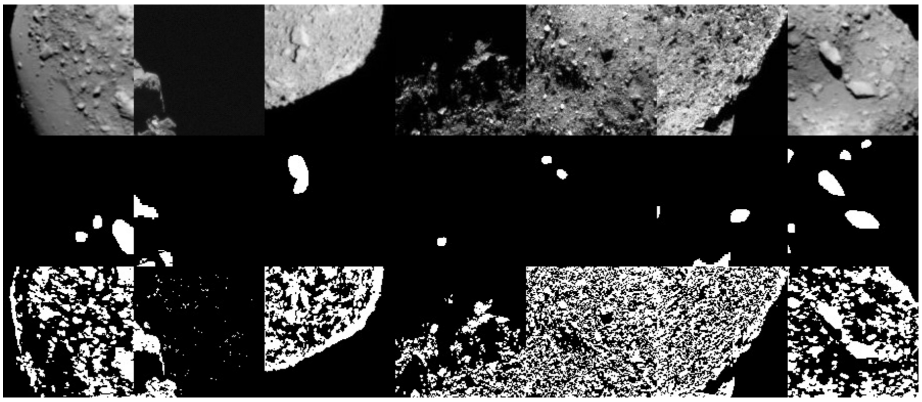


Figure 4.22: Samples of the input image (left), true mask (center), and predicted mask (right) from the UNet trained in $\textit{step}_4$ on test images of $\mathcal{DS}_2^{DS3}$.

labeling the prominent boulders in a limited set of 50 images from previously flown missions.

Contrarily to the UNet illustrated in Section 4.1, in this section, the training strategy put in place made it possible for the architecture to detect a larger boulder presence in the images correctly. The network can detect many more boulders than the one represented in the ground truth, making them unreliable. This ultimately proves a challenge for Bennu and Ryugu since there is a risk of predicting boulder fields as uniformly spread features all over the surface. Instead, the desired behavior would be somewhat in between being able to predict small-size boulders on the surface and distinguish them clearly from large, prominent ones embedded between them. This ultimately poses a challenging labeling problem that remains to be addressed.

4.3 Multi-scale segmentation with domain randomization

Building on errors and observations commented in the previous sections about the design of $\mathcal{DS}_1$ and $\mathcal{DS}_2$ and the impact they had on network performances, a multi-purpose, multi-scale labeled dataset for boulder segmentation and navigation about small bodies is briefly discussed in this section.

This dataset, referred to as $\mathcal{DS}_3$, is fully characterized in [141] and briefly described in Section A.1.3. Its focus is centered on addressing the issues reflected in network performance with real images highlighted in Section 4.1 and Section 4.2 while focusing on two crucial aspects: domain randomization and scaling.

Domain randomization means operating over the grayscale images so that variability can be introduced in the input data to design robust and reliable data-driven IP methods. The purpose of the dataset is thus not to represent the conditions of a specific navigation camera but rather to present to the algorithms a dataset in which camera noise, scattering properties, and illumination conditions are varied for each sample. This should enable the algorithms to become agnostic of these factors and instead learn fundamental geometrical relationships to be used for inference.

Scaling is explored by having the same regions of the image available at different resolutions and by exploiting the boulder's population at different sizes. Like domain randomization, the idea is to enable methods using this dataset to become scale-agnostic.

A sample of the dataset's image-label pairs is illustrated in Figure 4.23. $\mathcal{DS}_3$ is not limited to segmentation applications but is specifically designed for navigation purposes. For example, a few natural, easy-to-detect landmarks over the surface of a small body could be used onboard, substituting the human-in-the-loop approach illustrated in [60] while addressing the template-matching and illumination drawbacks of the NFT method illustrated in [61, 62]. For this reason, *15* prominent boulders scattered across the surface of the target body are identified with unique layers, while two layers identify the body's surface and all the smaller boulders.

In Figure 4.24, it is possible to see the relative frequency of appearance of each mask layer for the entire dataset. Both surface and minor boulders are always visible for each dataset sample, while each of the *15* prominent boulders appears with different frequencies. Note that Figure 4.24 represents two curves at the same time: the one with all the *47502* images of the dataset ("All") and the one that only focuses on the images labeled as *00000j_005* ("Only full"), as illustrated in Figure A.12. Such subdivision is performed since certain users (specifically those wanting to work on navigation with full views of the target asteroid) may be interested in using only such a subset of images. In contrast, others (e.g., those interested in segmentation) may instead be interested in using all images to increase the size of the training set.

In Figure 4.25, it is possible to see the relative and cumulative frequency of the appearance of multiple prominent boulders in the same image, which may be an interesting property for navigation purposes. Once again, the two cases of full dataset and full images are reported for completeness. The difference between the two is minor, with a natural tendency to have less prominent boulders visible in cropped images than in full ones.

Finally, in Figure 4.26, it is possible to see the histograms of the probability of appearance as a function of the number of pixels of the different layers illustrated in Table A.10. As expected, background and surface layers take a large portion of the image content, their areas being roughly one order of magnitude higher than that of boulder pixels. Furthermore, when considering differences between minor and prominent boulders, it is observed that the former occupies an area roughly one order of magnitude higher than the latter.

Albeit the dataset has yet to be used directly to train and test any data-driven IP algorithm, future works will use it to address the challenges and improvements highlighted in this chapter.

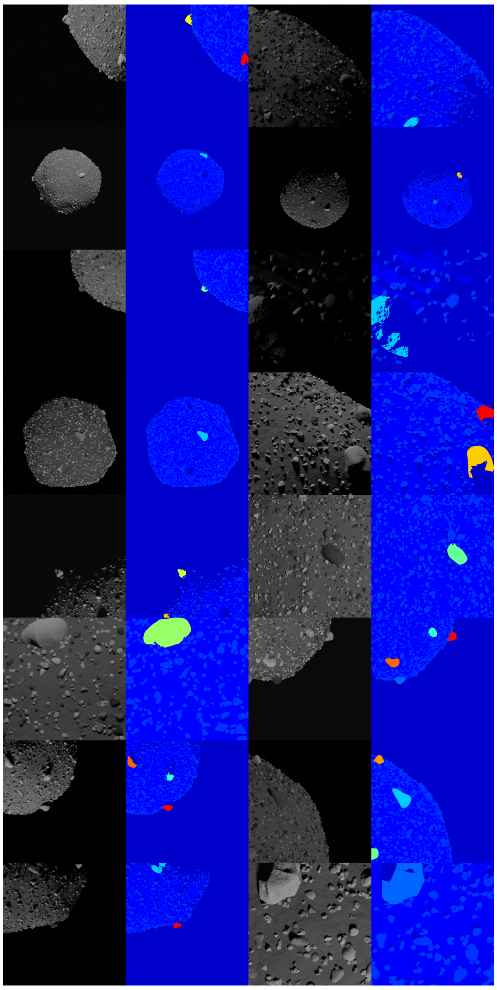


Figure 4.23: Mosaic view of a sample of the $\mathcal{DS}_3$ dataset with the input grayscale images and segmentation labels.

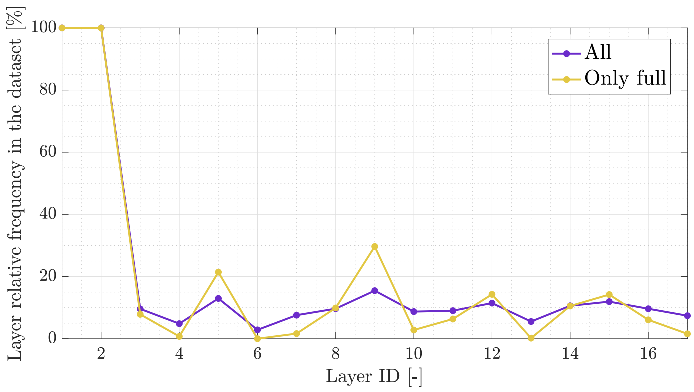


Figure 4.24: Relative frequency of appearance of each layer.

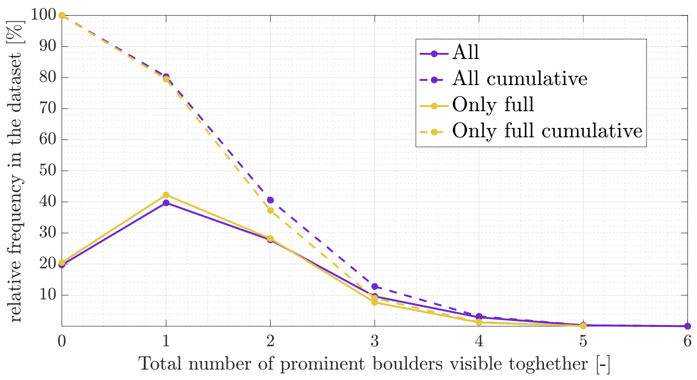


Figure 4.25: Relative (continue) and cumulative (dashed) frequency of appearance of prominent boulders.

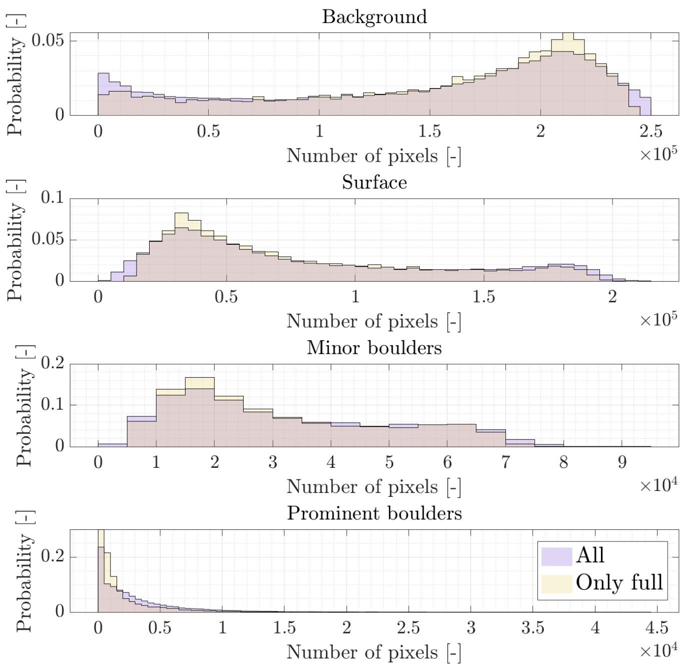


Figure 4.26: Probabilities of the different layers as a function of the area in pixel they each occupy in the masks. The binwidth in the histograms is *5000* samples for the first three subplots and *500* for the last.

4.4 Final remarks

In this chapter, an in-depth overview of the capabilities of DL networks for image segmentation tasks over the surface of small bodies has been extensively discussed. What follows is a list of final remarks.

- Image segmentation techniques that are correctly implemented and embedded onboard a spacecraft unlock the capabilities for various applications in the proximity of small bodies. These include but are not limited to:
  - **Autonomous scientific acquisitions**: Being capable of perceiving the presence of features of interest, such as craters and boulders, can be used onboard to autonomously control the spacecraft pointing and to perform autonomous smart acquisitions with scientific payloads. Specific features could be prioritized, and their presence in the sensor of a navigation camera can be used to derive a spacecraft's primary pointing. Increasing the system autonomy with smart scientific acquisitions would also reduce the overall data to downlink to the ground.
  - **Hazard detection**: The capabilities of robustly detecting large craters and boulders can increase the perception of the hazardousness of the terrain for landing applications, providing real-time capabilities for safe touchdown and landing site selections.
  - **Navigation**: The light-invariant and intensity-agnostic properties of segmentation masks can substitute onboard rendered templates of selected landmarks for feature-based navigation methods. This approach is further explored in the next section.
- A new methodology is developed in CORTO to generate automatically labeled datasets for the semantic segmentation of small bodies, exploiting the ray-tracing capabilities of Blender and using simple image processing methods. Such methodology is fundamental in allowing the training of DL methods in the image segmentation task, which is addressed in this chapter using UNet architectures.
- Three datasets have been used across the chapter: $\mathcal{DS}_1$, $\mathcal{DS}_2$, and $\mathcal{DS}_3$. A relevant domain gap between real and synthetic images is observed in $\mathcal{DS}_1$. This gap influences the performance of networks solely trained on synthetic samples and is observed to be particularly relevant for craters and boulders, whose modeling and photorealism are subsequently improved in the synthetic datasets of $\mathcal{DS}_2$ and $\mathcal{DS}_3$.
- The last iteration of the datasets presented, $\mathcal{DS}_3$, is a multi-scale labeled dataset tailored for boulder segmentation and navigation. It primarily focuses on addressing the issues reflected in network performance with real images highlighted in Section 4.1 and Section 4.2 while focusing on domain randomization and scaling. Instead of representing the conditions of a specific navigation camera, the dataset represents distribution samples of camera noise, scattering, and illumination properties used to generate the samples. Such properties should enable the algorithms using the dataset to become agnostic to these factors and instead learn fundamental geometrical relationships. Scaling is also explored by having the same regions of the image available at different resolutions (2x) and by exploiting the boulder's population with different sizes.
- Segmentation networks are successfully trained with bootstrap strategies

mixing large datasets of synthetic samples with small ones of real samples. This approach has proven efficient in making the network reach a higher plateau, shortening the training time, and improving generalization capabilities. The bootstrap strategy can also naturally be adapted for operations in real missions. The appearance of an unknown body can be assumed and represented with synthetic data to develop the bulk of network capabilities, which can then be tuned shortly after arrival using a limited amount of real samples collected in the first phases of a mission. Data augmentation is a crucial tool to increase the number of real samples for training. At some capacity, this approach has already been extensively used in OSIRIS-REx [61, 62] with an extensive collection of data before the touchdown event and can benefit real implementations of data-driven algorithms.

- By relying on well-characterized architectures trained with synthetic images, mission designers also have the option to weigh the contributions at architecture levels, mixing prediction from synthetic-based and real-based architectures. An example is showcased by the hybrid $\mathrm{UNet}_{\mathrm{H}}$ architecture, which combines its predicted output between twin architectures instantiated by different weights and biases. This hybrid approach could be useful in increasing the Technology Readiness Level (TRL) of DL applications, deploying networks with trusted capabilities that mission designers can confidently use in combination with unstable architectures.
- The specialization of the encoder for small body features did not bring relevant overall network performance advantages, indicating that larger, unspecialized networks can be used at the beginning of training.
- The ability of the network to generate an uncertainty metric that accompanies the predicted mask is explored as an additional skill. This could be useful in locating and isolating uncertain regions in the segmentation masks, eventually labeling the uncertain pixels as part of an extra layer. However, as these regions are often detected between class boundaries, the strategy implemented is observed to bring only marginal improvements in segmentation performance. The same has been observed when substituting the uncertain pixels with the second most probably class. As this capability is essential in increasing the confidence in DL methods for these applications, it should be considered a crucial step towards the operational application of these methods.
- Networks specialized in boulders segmentation are also investigated under various illumination conditions with single and multiple boulders. Both single and multiple boulders are robustly detected, but the latter exhibit challenging conditions in dense regions since their presence covers the surface almost entirely. Also, an incorrect behavior is observed when predicting boulders over the edge of the body. This happens as the ray-tracing algorithm in Blender correctly labels true boulders over the projection of the edge in the image plane, even when only a few pixels are observed. Such a labeling mishap may have promoted the network to classify the entire edge as a boulder, an unwanted behavior. The same phenomenon is not observed over the

terminator region.

- The masks associated with true images from previously flown missions in $\mathcal{DS}_1$ and $\mathcal{DS}_2$ exhibited a significant drawback. Since only the most prominent boulders are labeled in these masks, networks trained over synthetic images can detect many more boulders than the ones represented in the ground truth.
- Noise levels from real sensors do not pose particular challenges to the network regarding generalization. Albeit the network has been trained without a tailored noise setup modeling any specific camera, the network performs similarly in all types of images considered. It is thought that thanks to the injection of artificial noise and the particular care that has been put into the design of domain randomization strategies in the generation of the datasets, the networks developed generalization across a variety of changes in the input space.
- Complex training strategies have been investigated to take care of network learning. These often resulted in higher performance plateaus and better generalization capabilities, yielding overall better networks. These findings hint at network designers' need to approach training with incremental approaches using bootstrap strategies rather than end-to-end training.
- Finally, it is commented that network capabilities shall always be assessed with real datasets, as domain gaps with synthetic ones represent a significant source of errors for real applications. In this sense, the design of networks exploiting large synthetic datasets goes hand-in-hand with the quality and capability of such datasets to represent realistic conditions. Often, this can only assessed by inspecting network performance with real images, both qualitatively or quantitatively.

Navigation

"And, when you want something all the universe conspires in helping you to achieve it."

Paulo Coelho, *The Alchemist*

Relative optical navigation about a minor body is a challenge that can be tackled with different approaches. In this chapter, three navigation approaches around small bodies that make use of ML techniques are investigated. Navigation is addressed using regression, classification, and segmentation networks, considering both grayscale images or segmentation maps created with the networks illustrated in the previous chapter. Convolutional, correlation, and recurrent approaches are all investigated in this chapter to generate position estimates.

In doing so, two main frameworks are investigated: end-to-end architectures generating position estimates from input images and intermediate ones generating optical observables that need further (traditional) processing to generate position estimates. The former often uses centroid coordinates quantities expressed in the image plane and the predicted range to generate position vectors. Centroid coordinates are either represented as expected CoM coordinates (also referred to as CoF) or as correction vectors in the image plane between CoB and CoM (also referred to as δ).

Lastly, navigation approaches that use hand-crafted features or end-to-end strategies designed explicitly for benchmarking with the baseline IP of the Milani mission are presented in Section 6.6.

5.1 Convolutional architectures for navigation

This section investigates different strategies for using convolutional architectures to generate position estimates using single images of small bodies.

Four different types of architectures are investigated using various training strategies. To generalize network performance, four different small bodies representing different irregular shapes are chosen: 65803 Didymos ($\mathcal{D}$), 103P/Hartley ($\mathcal{H}$), 21 Lutetia ($\mathcal{L}$), and 67P/Churyumov–Gerasimenko ($\mathcal{P}$). For each of these bodies, five different labeling strategies that can be used to generate a position estimate are investigated. These are:

- **(δ,ρ)-$\mathcal{UV}$**: The estimated difference in image plane between the CoB and the CoM and the range extracted from images are used to generate a position vector in $\mathcal{CAM}$ frame, that can be transformed in $\mathcal{W}$ or $\mathcal{AS}$ by simulating onboard attitude determination from a star-tracker alongside the assumption of knowledge of the rigid body transformations. The transformation from image quantities to position vectors is the same as the one illustrated in Section 3.1.1.5. This strategy investigates how a feature vector with geometric quantities extracted directly from the image can generate a position vector in the $\mathcal{CAM}$ frame.
- **(ϕ_1,ϕ_2,ρ)-$\mathcal{AS}$**: The estimated position in polar coordinates around the target body expressed in the $\mathcal{AS}$ reference frame. This strategy investigates the image-label mapping using a reference frame fixed to the asteroid shape. The purpose is to examine if a mapping can be efficiently established by a network exploiting local (craters and boulders) or global (body's outline) features whose appearance varies under different illumination conditions.
- **(X, Y,Z)-$\mathcal{AS}$**: The estimated position in cartesian coordinates around the target body expressed in the $\mathcal{AS}$ reference frame. The aim of this strategy is similar to the previous one, with the main difference that spherical coordinates are investigated in substitution to cartesian ones. The former has the advantage of clearly decomposing the position estimate between cross-track and boresight axes, which could benefit a vision-based system.
- **(ϕ_1,ϕ_2,ρ)-$\mathcal{W}$**: The estimated position in polar coordinates around the target body expressed in the $\mathcal{W}$ reference frame. This strategy investigates the image-label mapping using a reference frame fixed to the illumination conditions. The purpose is to examine if a mapping can be efficiently established by a network exploiting local (craters and boulders) or global (body's outline) features whose appearance varies under different rotational states of an irregular body.
- **(X, Y,Z)-$\mathcal{W}$**: The estimated position in cartesian coordinates around the target body expressed in the $\mathcal{W}$ reference frame. The aim of this strategy is similar to the previous one, with the main difference that spherical coordinates are investigated in substitution to cartesian ones, with the expected benefits described before.

For improved handling of the labels, whenever the (ϕ_1,ϕ_2,ρ) labels are considered, the azimuth angle ϕ_1 is transformed in the adimensional ($sin\phi_1$,$cos\phi_1$) pair. The estimation vector is thus made of three elements for all labeling strategies but the (ϕ_1,ϕ_2,ρ) one, which comprises four elements.

For each target body, a dataset of 17500 images is generated with the five different labeling strategies and used for training, validation, and testing. More details about the dataset properties and generation methodology are discussed in subsection A.2.1.

The work presented in this section investigates three major questions regarding the usage of convolutional architectures for position estimation tasks: 1) While CNN are superior over complex sceneries, can simpler methods perform better when it comes to analyzing images of small bodies in the medium regime? 2) As suggested in [83], can a methodology be developed to bootstrap the training of CNN exploiting the capability of CELM to identify the most promising architectures? and 3) Which is the best labeling strategy and reference frame, and to what extent do they influence the performance and simplicity of training when considering the position estimation task around small bodies?

5.1.1 Training strategy

As illustrated in the previous section with the training of segmentation architectures and their encoders, a complex incremental training strategy is adopted, exploiting the fast training time of CELM to find optimal architectures to be trained as CNNs in a typical bootstrap strategy.

Hierarchical convolution architectures are generated following an established methodology to find the optimal set of Θ for each body and labeling strategy. A schematic of the overall procedure is illustrated in Figure 5.1 and is further described in detail in the rest of the section.

At first, many CELM architectures are initialized with a hierarchical structure as in [83], which is constructed procedurally. Going deeper from the input layer to the fully connected layer, the starting $128 \times 128 \times 1$ image is squeezed; its size s_i is halved as a function of the depth level i as $s_i = 2^{7-i}$ while its depth is doubled starting from an arbitrary value of 16 in the first layer. Each depth level i comprises the consecutive application of a convolution, activation function, and pooling operation. The convolutions are performed with 3×3 kernels while the number of kernels n_i^{ker} used at each depth is set to increase exponentially as:

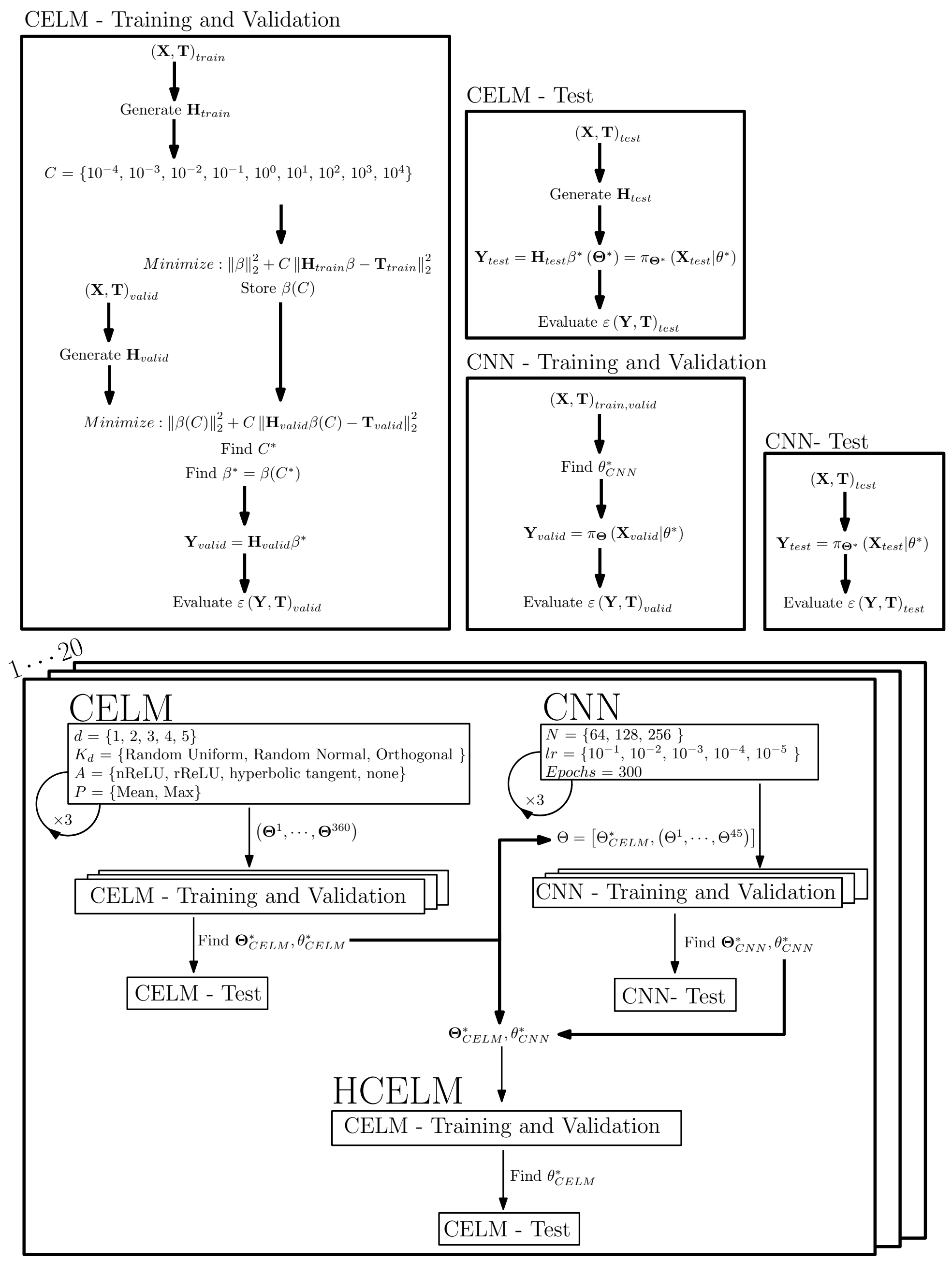
$$n_i^{ker} = 2^{3+i} \qquad n_0^{ker} = 1 \tag{5.1}$$

where i represents the depth level, such that $n_1^{ker} = 16$ is the starting depth in the first layer. Consequently, the number of weights and biases at each depth level of the encoder can be determined as follows:

$$n_i^W = 9 n_i^{ker} n_{i-1}^{ker} \quad , \quad n_i^b = n_i^{ker} \tag{5.2}$$

Network depths from 1 to 5 are explored. These correspond to a cumulative number of parameters in the convolutional portion of the networks, respectively of 160, 4800, 23296, 97152, and 392320. After the convolutional portion of the network, a fully connected layer is generated, whose size can be determined as:

$$n_i^{fc} = s_i^2 n_i^{ker} \tag{5.3}$$



CELM - Training and Validation
$(\mathbf{X},\mathbf{T})_{train}$
Generate $\mathbf{H}_{train}$
$C = \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}\}$
$Minimize: \|\beta\|_2^2 + C\|\mathbf{H}_{train}\beta - \mathbf{T}_{train}\|_2^2$
Store $\beta(C)$
$(\mathbf{X},\mathbf{T})_{valid}$
Generate $\mathbf{H}_{valid}$
$Minimize: \|\beta(C)\|_2^2 + C\|\mathbf{H}_{valid}\beta(C) - \mathbf{T}_{valid}\|_2^2$
Find C^*
Find $\beta^* = \beta(C^*)$
$\mathbf{Y}_{valid} = \mathbf{H}_{valid}\beta^*$
Evaluate $\varepsilon(\mathbf{Y},\mathbf{T})_{valid}$
CELM - Test
$(\mathbf{X},\mathbf{T})_{test}$
Generate $\mathbf{H}_{test}$
$\mathbf{Y}_{test} = \mathbf{H}_{test}\beta^*(\mathbf{\Theta}^*) = \pi_{\mathbf{\Theta}^*}(\mathbf{X}_{test}|\theta^*)$
Evaluate $\varepsilon(\mathbf{Y},\mathbf{T})_{test}$
CNN - Training and Validation
$(\mathbf{X},\mathbf{T})_{train,valid}$
Find θ^*_{CNN}
$\mathbf{Y}_{valid} = \pi_{\mathbf{\Theta}}(\mathbf{X}_{valid}|\theta^*)$
Evaluate $\varepsilon(\mathbf{Y},\mathbf{T})_{valid}$
CNN- Test
$(\mathbf{X},\mathbf{T})_{test}$
$\mathbf{Y}_{test} = \pi_{\mathbf{\Theta}^*}(\mathbf{X}_{test}|\theta^*)$
Evaluate $\varepsilon(\mathbf{Y},\mathbf{T})_{test}$
$1 \cdots 20$
CELM
$d = \{1, 2, 3, 4, 5\}$
$K_d = \{$Random Uniform, Random Normal, Orthogonal$\}$
$A = \{$nReLU, rReLU, hyperbolic tangent, none$\}$
$P = \{$Mean, Max$\}$
$\times 3$
$(\mathbf{\Theta}^1, \cdots, \mathbf{\Theta}^{360})$
CELM - Training and Validation
Find $\mathbf{\Theta}^*_{CELM}, \theta^*_{CELM}$
CELM - Test
CNN
$N = \{64, 128, 256\}$
$lr = \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$
$Epochs = 300$
$\times 3$
$\Theta = [\Theta^*_{CELM}, (\Theta^1, \cdots, \Theta^{45})]$
CNN - Training and Validation
Find $\mathbf{\Theta}^*_{CNN}, \theta^*_{CNN}$
CNN- Test
$\mathbf{\Theta}^*_{CELM}, \theta^*_{CNN}$
HCELM
CELM - Training and Validation
Find θ^*_{CELM}
CELM - Test

Figure 5.1: Schematic of the overall training, validation, and testing strategy adopted to find the best convolution architectures.

Similarly, the number of weights and biases of the head can be determined since the fully connected layer is directly connected with the output layer:

$$n_i^{\beta} = n_i^{fc} n_o \quad , \quad n_i^{\beta_0} = n_o \tag{5.4}$$

where n_0 is the number of neurons composing the output. Using this methodology, various architectures can be generated by varying hyperparameters Θ. Considering the combination of those illustrated in Table 5.1 (in particular d, K_d, A, and P) a total of *120* different hierarchical convolution architectures are generated. For each kernel distribution strategy, a random initialization is executed three times, effectively instantiating *360* training episode for each of the four bodies and five labeling strategies for a total of *7200* training instances.

Table 5.1: Sets of $\mathbf{\Theta}$ explored for CELM and CNN methods.

Parameter	Symbol	Description	Possible values
Number of layers	d	Number of hidden layers in the architecture	1, 2, 3, 4, 5
Kernel distribution	K_d	Random distribution of the weight and biases of the kernels	Random Uniform (-1,1), Random Normal (0, 1), Orthogonal
Activation function	A	Activation function used after the convolution operation	Normalized Rectified Linear Unit (nReLU), ReLU, tanh, none
Pooling strategy	P	Pooling strategy after the activation function	Mean, Max
Regularization coefficient	C	Regularization coefficient of Eq. (2.25)	10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}
Batch size	$\mathcal{B}$	Batch size used in the MBGD	64, 128, 256
Learning rate	lr	Learning rate used in the MBGD	10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}

A five layers architecture is illustrated in Table 5.2 for simplicity.

Table 5.2: Example of a five layers architecture with a three neurons output layer, made of *392′320* parameters (1.5 MB).

ID	Layer type	Output Shape	Parameters	Connections
I	InputLayer	($\mathcal{B}$, 128, 128, 1)	0	C1
C1	Conv2D	($\mathcal{B}$, 128, 128, 16)	160	A1
A1	Activation	($\mathcal{B}$, 128, 128, 16)	0	P1
P1	Pooling	($\mathcal{B}$, 64, 64, 16)	0	C2
C2	Conv2D	($\mathcal{B}$, 64, 64, 32)	4640	A2
A2	Activation	($\mathcal{B}$, 64, 64, 32)	0	P2
P2	Pooling	($\mathcal{B}$, 32, 32, 32)	0	C3
C3	Conv2D	($\mathcal{B}$, 32, 32, 64)	18496	A3
A3	Activation	($\mathcal{B}$, 32, 32, 64)	0	P3
P3	Pooling	($\mathcal{B}$, 16, 16, 64)	0	C4
C4	Conv2D	($\mathcal{B}$, 16, 16, 128)	73856	A4
A4	Activation	($\mathcal{B}$, 16, 16, 128)	0	P4
P4	Pooling	($\mathcal{B}$, 8, 8, 128)	0	C5
C5	Conv2D	($\mathcal{B}$, 8, 8, 256)	295168	A5
A5	Activation	($\mathcal{B}$, 8, 8, 256)	0	P5
P5	Pooling	($\mathcal{B}$, 4, 4, 256)	0	FC
FC	Flattening	($\mathcal{B}$, 4096)	0	O
O	Dense	($\mathcal{B}$, 3)	12291	

For each training instance, once the forward pass of the CELM is executed, the LS optimization problem is run nine different times with different regularization terms C. The training set is used to determine all possible values of β depending on C, while the validation set is used to determine the best value of $C$and is discussed in Section 2.2, is essential to avoid overfitting. Including the different values of Cconsidered from the validation sets, a total of $7200 \times 9 = 64800$ training episodes are therefore executed with this strategy.

By design, this large amount of training episodes are addressed using CELM theory, essentially using a single-shot regularized least square formula (Eq. (2.25)) instead of the iterative and time-consuming MBGD strategy for a conventional CNN. Because the training time of CELM is orders of magnitude faster than the one of the CNN ($\sim 1s$ compared to $\sim 600s$, on average for the task considered), with the use of CELM networks, it is possible to explore the architecture space to find those that seems inherently more suitable for the task at hand. This is motivated by the fact that a large portion of a network's performance appears to be attributed to its design [83], which is often neglected nor sufficiently explored given the long training time required by iterative GD methods.

For each dataset, given the navigation task at hand, the best CELM network is defined as the one achieving the minimum positioning error $\varepsilon^r_{\mathbf{p}}$ (defined in Section 1.8.5) on the test set while its parameters $\mathbf{\Theta}$ are saved. A CNN is then initialized with these hyperparameters and trained with a MBGD varying $\mathcal{B}$ and lr as per the values illustrated in Table 5.1, performing three runs for each combination of learning setup. This corresponds to *45* training instances for each body-labeling strategy and $45 \times 5 \times 4 = 900$ training episodes.

For each case, the CNN is trained for 300 epochs while the best value on the validation loss is used to instantiate the weights and biases of the best possible realization of the CNN to use at inference. Each training episode of the CNN takes roughly 600 s[31], fot a total of 7.5 hours for each body-labeling strategy considered and $7.5 \times 5 \times 4 = 150$ hours to train all the 900 CNN architectures. The set of $\boldsymbol{\Theta}$ found for the best CELM and CNN architectures is summarized in Table 5.3.

Table 5.3: Best sets of $\boldsymbol{\Theta}^*$ found during training and used in inference.

ID	Body	$\boldsymbol{\Theta}^*_{CELM}$					$\boldsymbol{\Theta}^*_{CNN}$	
		d	K_d	A	P	C	$\mathcal{B}$	lr
D1	$\mathcal{D}$	5	RandomUniform	tanh	Mean	10^{-1}	64	10^{-3}
H1	$\mathcal{H}$	5	Orthogonal	relu	Mean	10^{1}	256	10^{-4}
L1	$\mathcal{L}$	5	RandomNormal	tanh	Mean	10^{-2}	64	10^{-3}
P1	$\mathcal{P}$	5	RandomNormal	tanh	Mean	10^{-2}	64	10^{-3}
D2	$\mathcal{D}$	5	Orthogonal	nrelu	Mean	10^{1}	64	10^{-3}
H2	$\mathcal{H}$	5	Orthogonal	relu	Mean	10^{1}	64	10^{-3}
L2	$\mathcal{L}$	5	Orthogonal	relu	Mean	10^{1}	64	10^{-3}
P2	$\mathcal{P}$	5	Orthogonal	nrelu	Mean	10^{1}	64	10^{-3}
D3	$\mathcal{D}$	5	Orthogonal	none	Max	10^{-2}	64	10^{-3}
H3	$\mathcal{H}$	5	Orthogonal	nrelu	Mean	10^{1}	64	10^{-3}
L3	$\mathcal{L}$	5	Orthogonal	relu	Mean	10^{1}	64	10^{-3}
P3	$\mathcal{P}$	5	Orthogonal	nrelu	Mean	10^{1}	64	10^{-3}
D4	$\mathcal{D}$	5	Orthogonal	relu	Mean	10^{2}	64	10^{-3}
H4	$\mathcal{H}$	4	Orthogonal	relu	Mean	10^{1}	64	10^{-3}
L4	$\mathcal{L}$	5	Orthogonal	nrelu	Mean	10^{1}	64	10^{-3}
P4	$\mathcal{P}$	5	Orthogonal	nrelu	Mean	10^{1}	64	10^{-3}
D5	$\mathcal{D}$	5	Orthogonal	nrelu	Mean	10^{1}	64	10^{-3}
H5	$\mathcal{H}$	5	Orthogonal	relu	Mean	10^{1}	64	10^{-3}
L5	$\mathcal{L}$	5	Orthogonal	relu	Mean	10^{1}	64	10^{-3}
P5	$\mathcal{P}$	5	Orthogonal	relu	Mean	10^{1}	64	10^{-3}

Two additional hybrid setups are also investigated, referred to as Hybrid Convolutional Extreme Learning Machine (HCELM) and Hybrid Convolutional Extreme Learning Machine 3 (HCELM$_3$). In HCELM, transfer learning is used to combine the weights and biases of the kernels from the encoder of a previously trained CNN architecture with a new training episode that interests only the head of the network, which is trained using the LS method as in the CELM paradigm. Similarly, HCELM$_3$ considers as initial architecture for training the best-performing one given a specific reference frame (among the $\mathcal{UV}$, $\mathcal{AS}$, and $\mathcal{W}$ reference frames) and

[31] Using a Tesla P100-PCIE 16Gb GPU, with a 27.3 Gb of RAM in Google Colab https://colab.research.google.com/, retrieved 13th of September, 2023.

then tune its last layer using the LS method, as in HCELM. In Figure 5.2, the schematic difference between the architectures used in this section is illustrated. It is highlighted that CELM, CNN, HCELM architectures share the same global structure, populated by different weights and biases.

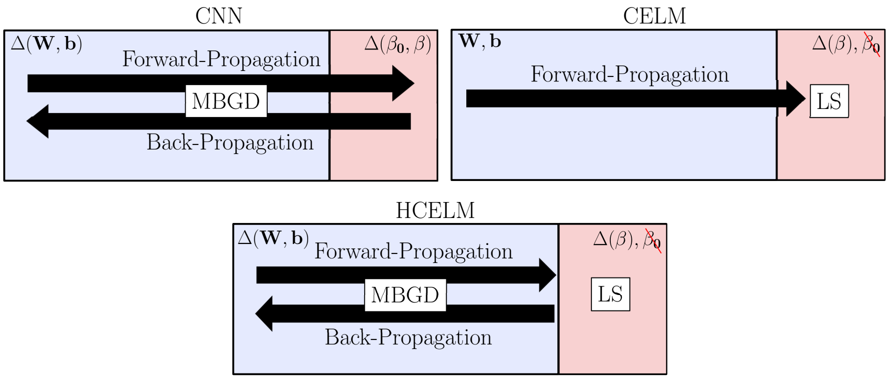


Figure 5.2: Schematic of the main differences between CELM, CNN, and HCELM architectures.

All four architectures are considered to work with normalized input and output, which has been observed to improve the overall performance. While CELM are trained with the entire dataset at once, CNN architectures are trained using batches $\mathcal{B}$. In the case of CNN architectures, this was due to hardware limitations, while the number of *7500* images for the training set similarly comes from hardware limitations related to matrix inversion in the CELM training. Adam is used as an optimization algorithm to train the methods requiring MBGD.

5.1.2 Results

The four architectures, referred to as CELM, CNN, HCELM, and HCELM$_3$, are trained over all datasets of $\mathcal{DS}_4$ with the set of hyperparameters Θ detailed in Table 5.3.

At inference, they are applied over the test sets of $\mathcal{DS}_4$, and their performance is characterized using the metrics defined in Section 1.8.1 and Section 1.8.5. For simplicity, a shared colormap is used in this section to distinguish between the four methods: Purple, blue, green, and yellow are used respectively for the CELM, CNN, HCELM, and HCELM$_3$ architectures.

5.1.2.1 Global

From Figure 5.3, Figure 5.4, and Figure 5.5 global performance in terms of $\varepsilon^r_{\mathbf{p}}$ are summarized for all cases considered. From these plots, it is possible to draw important considerations on the effects of labeling strategy, reference frame, and training strategy.

In Figure 5.3, the box plot is organized from top to bottom in macro-groups of five (depending on the labeling strategy adopted) and then in clusters of four (based on the network). The reader can also consult Table A.12 to understand the reference frame, body, image processing method, and labeling strategy adopted for each value represented on the vertical axis of Figure 5.3.

Focusing on the influence of the reference frame and labeling strategy (δ, ρ) outperforms all other approaches considered by a large margin. This could be motivated by the fact that being the labels δ, ρ quantities that can be directly extracted from images they make it simpler for the networks to learn relatively straightforward image-label representations. It is also observed that better performance is achieved using $\mathcal{W}$ rather than $\mathcal{AS}$. This hints at the significant impact of illumination conditions on learning a target body representation within a network. The size of the training set could also cause this; increasing it, the performance of $\mathcal{AS}$-based networks may improve. Nonetheless, this points toward the $\mathcal{AS}$ labeling as being the most inefficient form for position estimation between the two. Lastly, the choice of the coordinate systems (cartesian or polar) is observed to play a minor effect on performance.

Considering the (δ, ρ) strategy, it is remarkable that the CELM method performs similarly to all other networks, which is not observed in different labeling strategies. Once again, this hints at the simple representation form of this labeling strategy, which seems approachable with simple approaches. Overall, CNNs outperforms all architectures for each body and labeling strategy considered.

A more concise representation is reported in Figure 5.4. Once again, it is possible to appreciate the much better performance achieved with the (δ, ρ) labeling strategy than the others for all bodies considered. In such a case, the performance of CELM is similar to that of the CNN. While the former has a mean $\varepsilon_{\mathbf{p}}^{r}$ of 2.58, 8.63, 5.60, 5.02 (respectively for $\mathcal{D}$, $\mathcal{H}$, $\mathcal{L}$, and $\mathcal{P}$), the latter generates position estimates only 1.68, 4.11, 1.68, and 1.63 times better. Apart from $\mathcal{H}$, this means that only a very marginal performance gain is achieved with the use of a CNN rather than a CELM. This does not hold when comparing CELM and CNN performances for other labeling strategies, which show much wider gaps.

From Figure 5.4 it is also possible to observe a trend depending on the shape considered: simpler, regular shapes such as the one of $\mathcal{D}$ are better exploited for navigation than highly irregular ones such as $\mathcal{L}$, $\mathcal{P}$ and $\mathcal{H}$.

Figure 5.5 reports in a stacked histogram plot the share for each dataset associated with the best IP method. While the CNN is always the best candidate, HCELM consistently scores as the second-best method across all datasets considered. Moreover, the CELM is considered the best third option only in the case of the (δ, ρ) labeling strategy. This graphic representation is possible because the geometric points considered across the test sets are the same.

Finally, Figure 5.6 reports metrics assessing the impact on the global positioning error caused by errors in the boresight direction. To do so, the metric $\mu(\frac{\varepsilon_{\rho}}{\varepsilon_{\mathbf{p}}}100)$ is defined to represent the mean percentage of the whole positioning error attributed

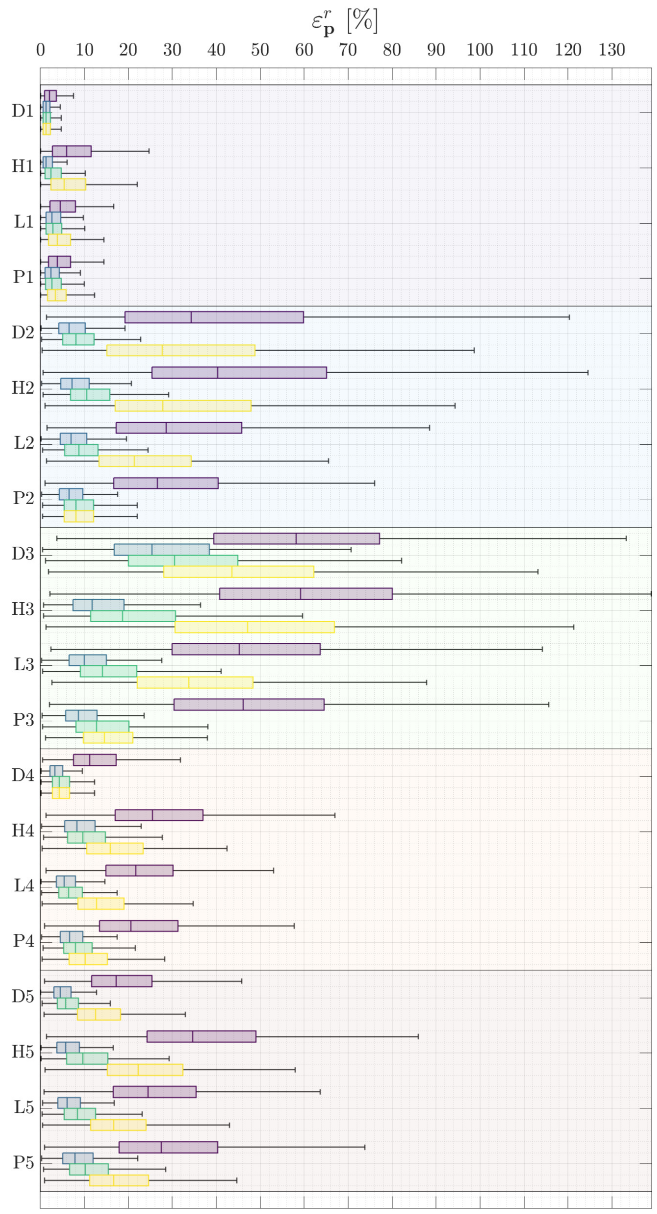


Figure 5.3: Box plot of $\varepsilon^r_{\mathbf{p}}$ for all cases considered.

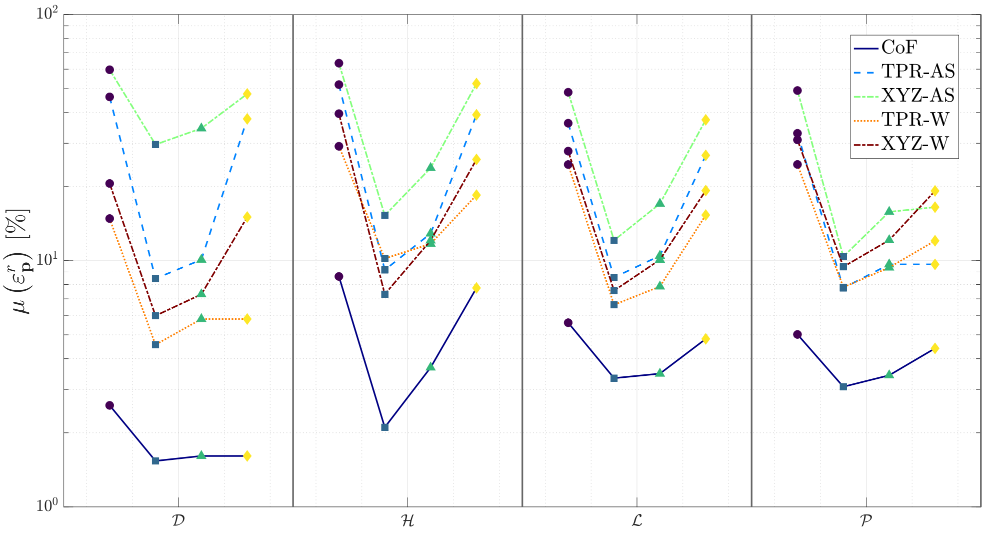


Figure 5.4: Summary plot of the mean $\varepsilon_{\mathbf{p}}^r$ values achieved by all possible combinations of the dataset and architecture investigated in this section.

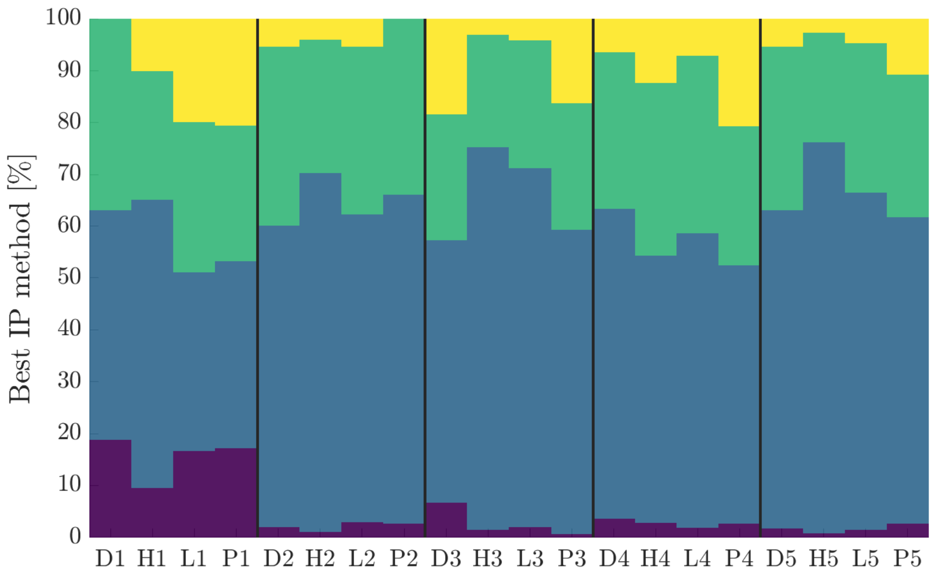
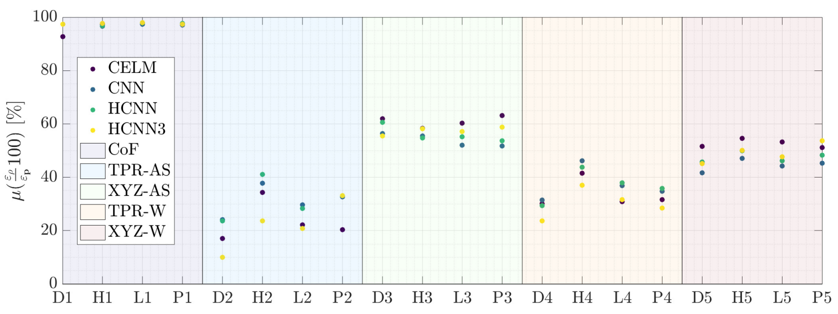


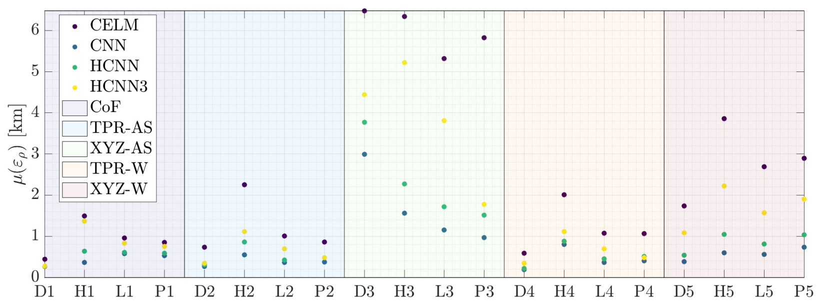
Figure 5.5: Shares of the best IP method across different datasets.

to the error in the boresight direction. Representing this metric in Figure 5.6(a), it is possible to observe different values corresponding to the different cases. In particular, it is possible to observe that the performance is heavily influenced by the adopted labeling strategy.

In the (δ, ρ) strategy, the error in the radial direction contributes almost entirely



(a) $\mu(\frac{\varepsilon_\rho}{\varepsilon_\mathbf{p}}100)$ for different cases.



(b) $\mu(\varepsilon_\rho)$ for different cases.

Figure 5.6: Plots of the $\mu(\frac{\varepsilon_\rho}{\varepsilon_\mathbf{p}}100)$ (a) and $\mu(\varepsilon_\rho)$ (b) values for the different cases considered.

($> 90\%$) to the entire positioning error. Significant contributions are also observed whenever cartesian coordinates are used ($40 - 60\%$). Lastly, whenever spherical coordinates are used, a small contribution is observed ($< 40\%$).

It is important to note that the relative contributions illustrated in Figure 5.6(a) must be put into perspective with absolute positioning performance, as the ones illustrated in Figure 5.6(b) and in Figure 5.3. For example, even though the percentage of the error in the radial direction is very high in the (δ, ρ) strategy, the total positioning error is very low compared to the other strategies. At the same time, it is observed that the range is reconstructed with similar performance whenever using the (δ, ρ) or spherical coordinates, as both strategies explicitly list the range from the target body among the components to estimate.

5.1.2.2 (δ, ρ) labeling strategy

Since the labeling strategy based on (δ, ρ) seems the best performing one, the navigation performance of this strategy is further addressed in this section.

In Figure 5.7 the histograms of the ε^n_{CoF} and ε_ρ metrics in $\mathbb{S}_2$ space are illustrated. When considering $\mathcal{D}$, the CELM shows a much larger mean error in ε^n_{CoF} than the other methods while only a smaller variability both in terms of means and variance is observed in ε_ρ.

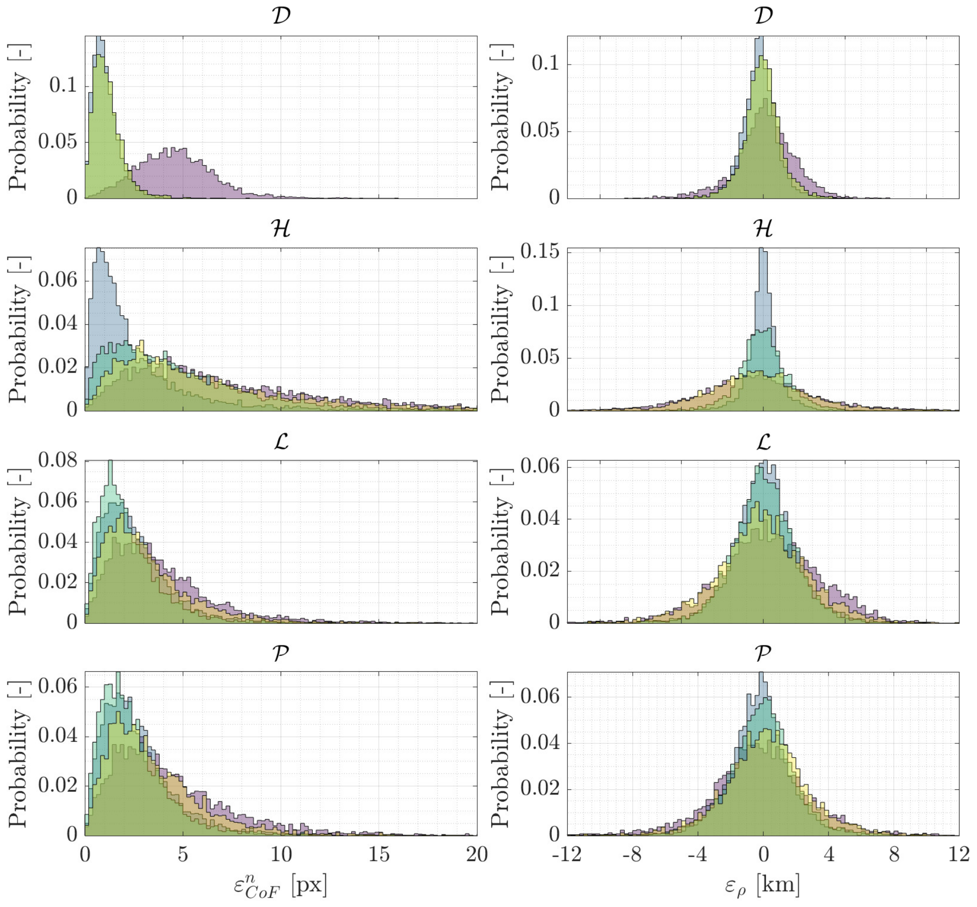


Figure 5.7: Normalized histograms of the ε_{CoF} (left) and ε_ρ (right) on different bodies with IP methods considered. Binwidth is *0.2* px for ε_{CoF} and *0.25* km for ε_ρ.

The fundamental failure mechanism for which the CELM method performs worse than the CNN for this body is caused by a large mean error in the estimated CoF coordinates. When considering $\mathcal{L}$ and $\mathcal{P}$, the difference between the histograms of the various methods is more subtle since only minor variations in the mean and variance are observed across the different methods. On the other hand, when considering $\mathcal{H}$, it is possible to see that the CNN is much more accurate both in the CoF and range estimates than all other methods. The hybrid CELM performs better in the range estimate (with similar variance than the CNN) but does not perform at the same level as the CNN in the CoF estimate.

In Figure 5.8, the error ellipses of the CoF coordinate in $\mathbb{S}_2$ spaces are illustrated together with the error ellipse (dashed dark ellipse) that would have been obtained by not correcting the CoB with a data-driven scattering law implemented by the IP. It is also observed that differently than all other ellipses, the CELM ellipse with $\mathcal{D}$ is not centered in the zero error point, thus introducing a bias that is causing the larger mean error already observed in Figure 5.7.

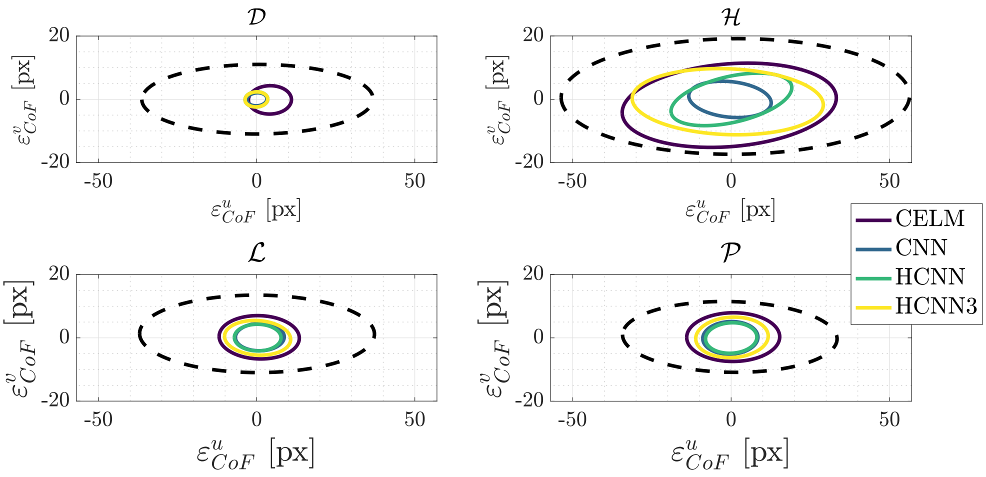
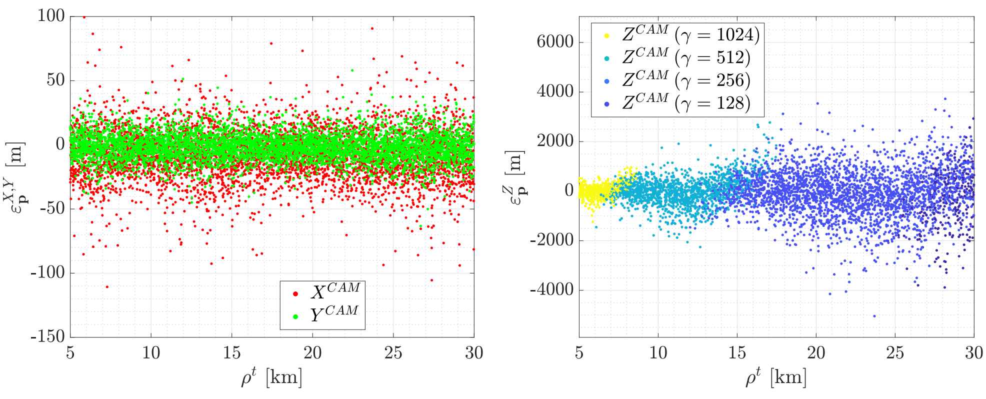


Figure 5.8: Error ellipses in $\mathcal{UV}$ frame and $\mathbb{S}_2$ space on different bodies with IP methods considered.

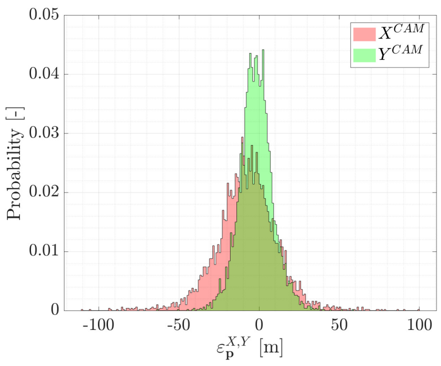
In Figure 5.9, a representative case of the position error of the CELM in $\mathcal{CAM}$ reference frame for the $\mathcal{L}$ body is illustrated with the CNN method. The error in the boresight direction is one order of magnitude higher than on the other axes. This result is expected from optical-based navigation systems. Therefore, the error in the range estimate from the body is the major contribution to the error in the position estimate when using this labeling strategy. Since most of the error comes from the radial direction, including the attitude error from a star-tracker, performance is not expected to vary significantly.



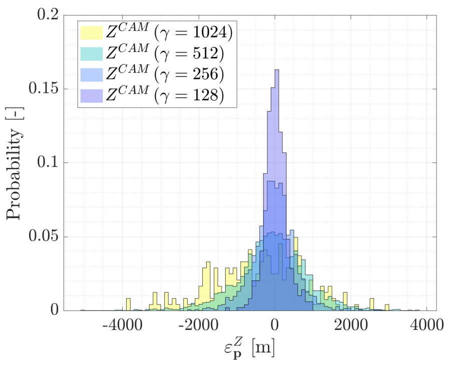
(a) XY coordinates. **(b)** Boresight direction.

Figure 5.9: Position error in $\mathcal{CAM}$ reference frame in XY coordinates (a) and boresight direction (b).

Lastly, a visualization of a sample of *200* images from the test set in $\mathbb{S}_2$ and the CoF prediction by the different IP methods is illustrated in Figure 5.12.



(a) XY coordinates.



(b) Z coordinate.

Figure 5.10: Histograms of the position error in $\mathcal{CAM}$ frame. XY coordinates (a) and boresight direction (b). The binwidth of both histograms is *100*m.

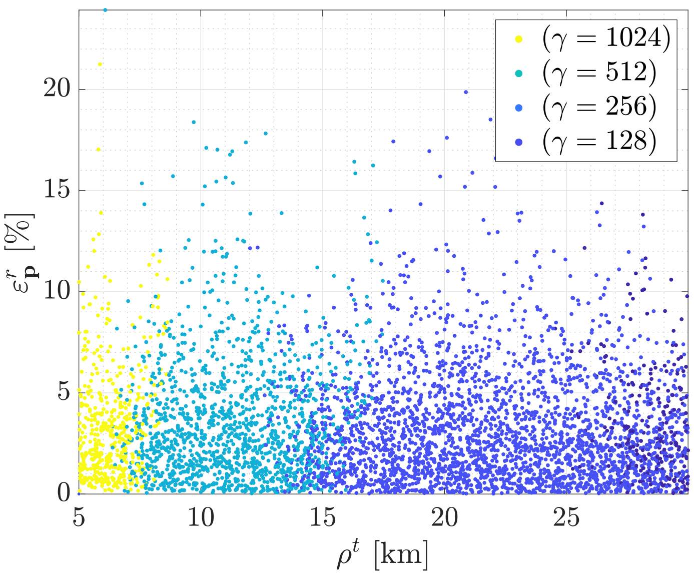


Figure 5.11: $\varepsilon_{\mathbf{p}}^{r}$ as function of ρ^{t} with the different values of γ used in the postprocessing to pass from $\mathbb{S}_0$ to $\mathbb{S}_2$.

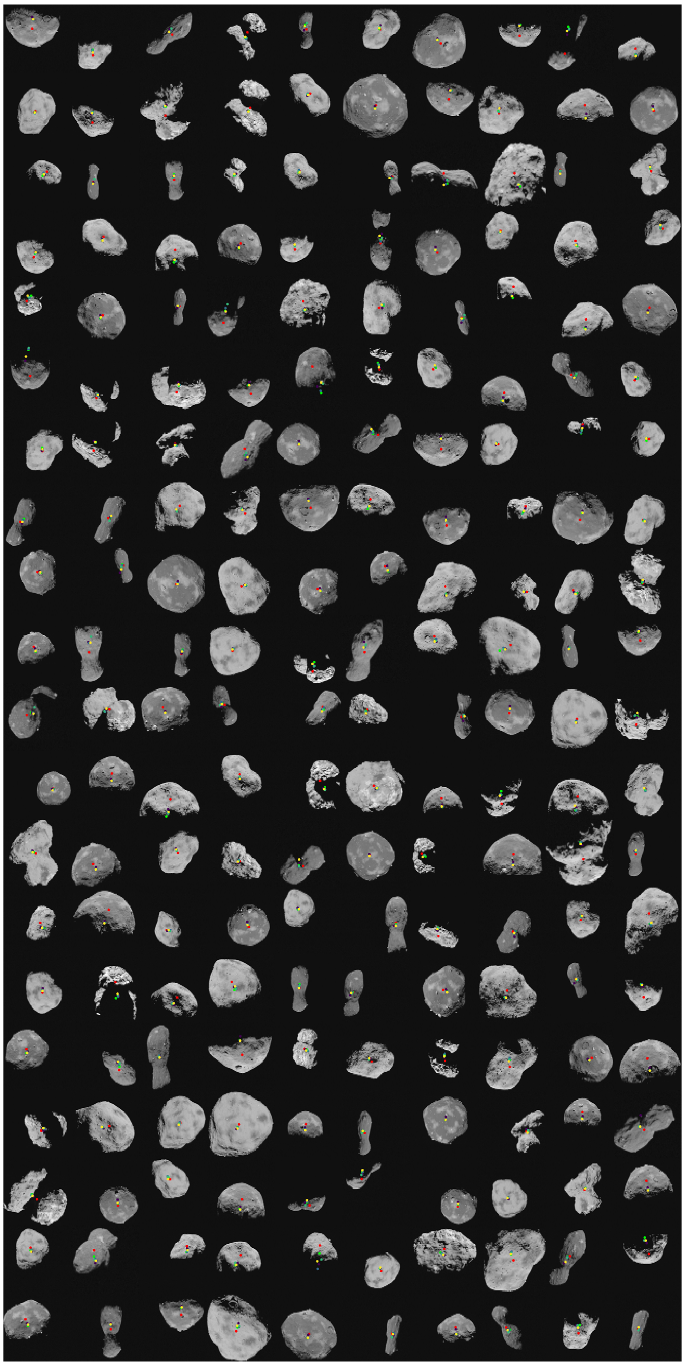


Figure 5.12: Sample of CoF estimates on images from the test set of $\mathcal{DS}_4$ with CoB (red) and CoM (green) visualized.

5.2 Segmentation and Navigation

Correlation methods are often employed to reconstruct the pose of a camera by attempting to match a real observation with a synthetic one, representing the body outline or local patches of the surface depending on the relative distance. The synthetic observation, often called a template, is generated via a model rendered onboard (but in some cases [60] it can also be rendered on-ground) via an artificial environment that controls the relative body-camera-Sun poses. Because these poses are affected by errors, multiple combinations are rendered simultaneously. Thus, the more complex the model and rendering procedure adopted, the more computationally expensive it is to run the technique onboard.

A significant source of errors in correlation methods is caused by the artificial simulation of the interaction between the surface material and illumination conditions, which are challenging to replicate in a fast way onboard a spacecraft using an artificial environment, especially considering the simultaneous generation of multiple templates. Illumination conditions have a strong influence over pixel-intensity-based metrics used for correlation when considering grayscale visual images. Moreover, it may be necessary to initialize correlation methods in a convergence basin as close as possible to the actual solution.

Exploiting segmentation maps such as the one illustrated in chapter 4, it would be possible to overcome some limitations while reducing the computational complexity. The advantage of using segmentation maps in place of grayscale images is twofold: simple, low-resolution models can be considered for online rendering, and at the same time, pixel-intensity variations due to illumination conditions do not play a direct role in the correlation.

In the following sections, a methodology is explored that uses segmentation maps and a correlation scheme for navigation purposes. The proposed architecture employs a hybrid segmentation-classification-regression scheme for two target bodies: $\mathcal{D}$ and $\mathcal{H}$, chosen to be representative of regular and irregular minor bodies.

5.2.1 Architecture

An onboard architecture is proposed to estimate a spacecraft's position taking as input grayscale visual images. The architecture is divided into three portions, as illustrated in Figure 5.13.

The first portion addresses the segmentation task using a UNet architecture as illustrated in chapter 4. The purpose of this portion is to transform grayscale pixel intensity to semantic meaning, deconstructing image pixel content into categories, which can later be used for navigation. The classes considered by the proposed architecture are background, surface, boulders, and craters, similar to those illustrated in Section 4.1. In the second portion, a rough position estimate is generated by solving a classification task with a CNN. The classifying network's primary purpose is to significantly reduce the parameter's search for the subsequent step. The third and last portion performs a discrete classification using an algorithm based on a Normalized Cross Correlation (NCC) metric. This section refines the rough solution provided by the CNN and is necessary to increase the accuracy of the architecture.

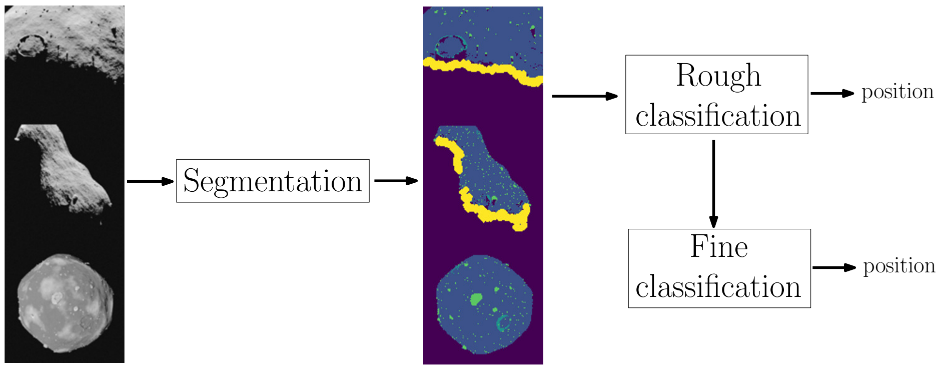


Figure 5.13: Macro-steps of the proposed method for small-body navigation with segmentation maps.

Classification has been preferred over regression to generate rankings between the predicted classes based on their correlation scores. This was motivated by the desire to investigate class distributions in the predictions and have a smaller dataset not to disperse the position labels. The CNN used to solve the classification task performs a twofold task. First, as a robust navigation method that can roughly locate the spacecraft around the body. Second, as a proxy of an onboard rendering engine, since its parameters embed the appearance of the segmentation maps used to represent the target body in a variety of geometric and illumination conditions.

Previous works also preferred classification over regression. For example, in [142], the position of a spacecraft with respect to the Moon is estimated using a CNN trained over images distributed across different classes over a digital terrain map. In [143] an interesting geolocation application is illustrated using a dynamic resolution grid of classes around Earth's surface to locate an image depending on its content. An interesting point illustrated in [143] is the possibility of grouping images that correlate the most due to their similarity, even from different geographic locations.

The classification framework, however, has one major drawback: by design, it cannot achieve high accuracy, which is tied to the total number of classes considered, their size, and the amount of data needed to represent them. Each class size bounds the maximum error, posing a significant limitation in performance that is overcome easily with the adoption of the subsequent NCC step.

The regions considered for classifications are illustrated in Figure 5.14. A total of *1176* volumes are carved for each body around a spherical shell in the $\mathcal{AS}$ reference frame. Each region is defined in polar coordinates by intervals of *15* deg for the equatorial and azimuth angle and *0.1* Blender Unit (BU) intervals for the range. For clarity, only the innermost and outermost of the *1176* classes are illustrated in Figure 5.14.

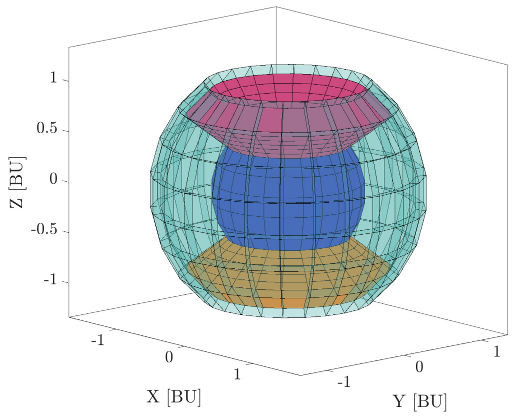


Figure 5.14: Structure of the 1176 classes distributed over the spherical shell. The blue and cyan are the innermost and outermost classes from the small body. The red and orange are the top and bottom classes with the maximum elevation value. In between, the other classes are not shown for clarity.

5.2.1.1 Rough classification

The classification task is addressed using a CNN. The hyperparameters search has been divided into two parts following an approach highlighted in [144] and adopting the philosophy described in [76]. In the first one, without any regularization in place, an optimal set in the parameter space for the learning rate, batch size, $\mathcal{B}$, and depth of dense layers is performed for a concise amount of epochs. After optimal regions of these parameters are identified, epochs are increased, and regularization is introduced in the form of dropout rates and image shifts. This process is repeated by refining the parameter space alongside reducing the number of cases and increasing the training epochs. The whole procedure considers global metrics for both training and validation accuracy as well as the convergence speed. The most promising ensembles of the network are then fine-tuned.

The same architecture is used for Didymos $\mathcal{D}$ and Hartley $\mathcal{H}$ and uses as input segmentation maps generated in CORTO with the methodology described in Section A.2.2, generating the dataset $\mathcal{DS}_5$. The weights and biases used are the ones achieving maximum accuracy on the validation sets. The outcome of the hyperparameter tuning for the best architecture is summarized in Table 5.4, while the architecture is illustrated in Table 5.5.

The standard definition of accuracy turned out to be limiting in comprehensively describing the more complex behavior of the classifier. In the specific classification task considered, spatial proximity between classes is essential but not encoded with the class label. In a traditional classification task, the classes may be defined by different objects, animals, or instances of a larger group, but they would still exhibit characteristic features that would make them different and distinguishable. However, in the task considered, classes next to each other do not represent different species

Table 5.4: Hyperparameters of the CNN architectures used in this section.

Parameter	**Value**
Optimizer	Adam
Loss function	Sparse Categorical Cross Entropy
α (MobileNetV2)	0.35
Dropout rate (1)	0.3 (Didymos), 0.15 (Hartley)
Dropout rate (2)	0.5
Learning rate	0.0005
β_1	0.9
β_2	0.999
Pixel shift	5
Batching strategy	mini-batch
Batch size $\mathcal{B}$	128
Steps per epoch	339
Training epochs	100

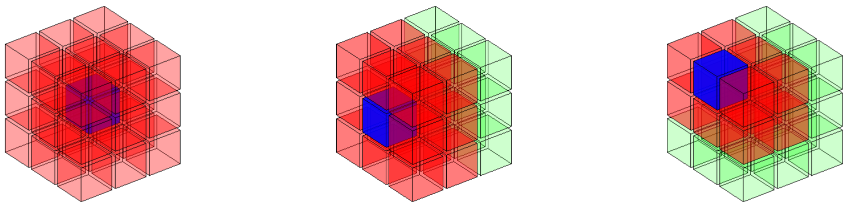
Table 5.5: Architecture of the CNN considered for classification. The total number of parameters is 725'880 (2.8 MB), of which 14'080 are not trainable.

ID	Layer type	Output Shape	Parameters	Connections
I	InputLayer	($\mathcal{B}$, 128, 128, 1)	0	F1
F1	InputTransform	($\mathcal{B}$, 128, 128, 3)	0	F2
F2	MobileNetV2$^{128}_{0.35}$	($\mathcal{B}$, 4, 4, 1280)	410208	A1
A1	Global average	($\mathcal{B}$, 1280)	0	DO1
DO1	Dropout	($\mathcal{B}$, 1280)	0	D1
D1	Dense	($\mathcal{B}$, 128)	163968	DO2
DO2	Dropout	($\mathcal{B}$, 128)	0	O
D1	Dense	($\mathcal{B}$, 1176)	151704	

of objects but the same object image from a slightly different perspective and with varying illumination conditions. Thus, the case of a wrong classification next to the correct class shall not be easily dismissed as a completely wrong classification since it indicates the encoder's capability to provide a wrong solution spatially close to the correct one. A metric considering the proximity between classes is therefore designed to account for this distance.

The Inter-Class Distance (ICD) is defined as the minimum number of classes that shall be passed through that connect a given point P in class A with another point Q in class B. A movement from one class to another is allowed if the two share a face, an edge, or a vertex. To illustrate how the ICD works, it is easier to think of it when considering the classes distributed over a 3×3 cube (as a classical Rubik's cube), as illustrated in Figure 5.15.

Such a cube is composed of *27* smaller cubes that can represent classes. Consider now a point in the center class (where the spherical joint of the Rubik's cube is hidden). From this point, each of the remaining *26* classes can be reached with a single jump between them. Because of the characteristic topology of the spherical shell of classes considered, only one of the *3* scenarios in Figure 5.15 is encountered when starting the count of the number of jumps between classes. From each of the *1176* classes of the spherical shell considered for classification, the ICD is evaluated with all the other $1176-1$ classes and stored in an upper triangular matrix.



(a) Central class. **(b)** Lateral class. **(c)** Edge class.

Figure 5.15: Various examples to illustrate the concept of ICD from the blue cube: blue (ICD0), red (ICD1), green (ICD2). (a) central class surrounded by 26 ICD1. (b) lateral class surrounded by 17 ICD1 and 9 ICD2. (c) edge class surrounded by 11 ICD1 and 15 ICD2.

The classification accuracy is then defined considering the ICD metric. The $a[ICDn]$ is the accuracy over the classes with an ICD equal to n while the notation $a[ICDn+]$ and $a[ICDn-]$ are used for the cumulative accuracy, taking into account all classes respectively above or within a distance of $ICDn$.

5.2.1.2 Fine classification

In a correlation procedure, an image is taken as a sample from the real environment and confronted with a set of templates saved in a database or rendered onboard. Controlling the parameters generating the templates (mainly camera-body-Sun poses and material properties), it is possible to iteratively maximize the similarity between the actual image and the synthetic templates. The template maximizing the similarity can then be used to estimate the relative pose of the camera. Different metrics can be used to assess this similarity. In the architecture presented in this section, a correlation metric derived from [145] is adopted:

$$\gamma = \frac{\sum_{ij}(R_{ij} * T_{ij})}{\sqrt{\sum_{ij} R_{ij}^2 * \sum_{ij} T_{ij}^2}} \tag{5.5}$$

Since segmentation maps are used for correlation, the albedo difference between the template and the real map does not play a direct role. It could do so indirectly if the segmentation method used is not robust enough to varying illumination conditions. For this reason, the formula in [145] has been modified to avoid such sensitivity. Note that the correlation metric is computed in the spatial domain and not in the frequency one for practicality.

Assuming the correlation is executed onboard, the model used for the template renderings is simpler than the one used to emulate the real segmentation maps processed from images taken in the environment. In practice, this is obtained by relaxing the rendering settings, changing the material properties and the mesh structure, removing the terminator layer, and considering only the biggest boulders identified in the images. For comparison, a rendering of the real segmentation map takes roughly 15s, while the rendering of the simplified model is executed in 0.3s considering a Cycles CPU rendering in Blender [32]. This test has been performed to understand the gain in computational time with respect to the rendering of full masks and is not representative of the computational time required on a space-qualified processor.

Other assumptions that have been taken are that the illumination conditions are known and that the correlation search space is only driven by variations in $R-\theta-\phi$ components. The pointing is assumed to be ideal, even though the CNN can accommodate non-ideal pointing.

With the described setup, an iterative correlation algorithm is used. As a starting point, the predicted class from the CNN and all the classes immediately next to it are used to define the initial intervals of the search space. A random distribution of an arbitrary set of 100 points is then generated in polar coordinates within this region, and the correlation coefficients between real and template maps are computed using Eq. (5.5). The point with the highest correlation is saved, and a new iteration is performed by reducing the search space interval of each component to $1/3$ of the original one. The new interval is used, and the procedure is repeated for three iterations, ensuring that the point with the absolute best coefficient survives from one iteration to the next. An example of a real segmentation map (also containing small boulders) and the subsequent onboard-generated segmentation ones during the three iterations of the correlation procedure are represented in Figure 5.16.

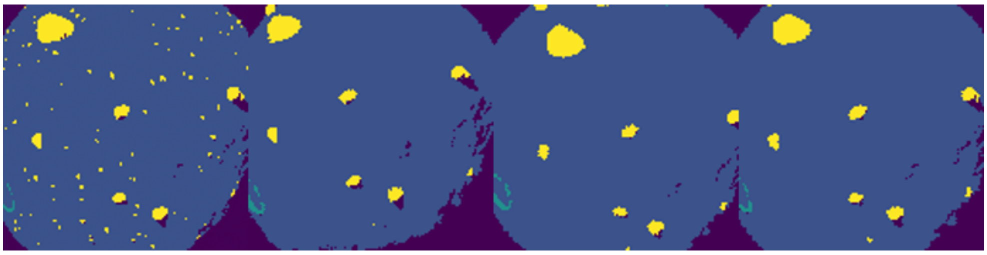


Figure 5.16: Comparison between the real segmented maps (Left) and the template ones with maximum correlations at the first three iterations. The correlation coefficient varies from left to right from 0.58, 0.69, and 0.75, while the error from 281, 111, and 21 m, respectively.

Variants of the algorithm with different interval reduction strategies, in Cartesian coordinates, and pixel-intensity-based metrics have also been tested but did not yield significant improvements.

[32]Performed using an Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz

Whenever an image is taken from a distance larger than $1.35D_0$ (D_0 representing the distance at which the body saturates the sensor's FOV), a series of simple image processing tasks is performed to transform the image appearance to one suitable for the CNN. As illustrated in Section 3.1.1.5, cropping, down-sampling, and re-scaling are used. The primary assumption underlying this is that apart from scaling, the appearance of the bodies does not differ significantly due to perspective changes given by different ranges-FOV configurations. This is necessary since the CNN is only trained within a limited envelope, represented by the 1176 classes illustrated in Figure 5.14.

5.2.2 Results

This section illustrates the results accomplished by the architecture on the four test scenarios detailed in the $\mathcal{DS}_5$ dataset described in Section A.2.2 and whose properties are reported in Table A.14. For consistency, the results of an additional scenario about Didymos, $\mathcal{DS}_5^{D-4}$, are presented in Section 6.6.2 instead.

For simplicity, in all datasets but in $\mathcal{DS}_5^{D-3}$, the segmentation maps used are the true ones, while in $\mathcal{DS}_5^{D-3}$ the predicted ones with the UNet described in [134] are used. The CNN performances over classification are assessed with datasets $\mathcal{DS}_5^{D-1}$, $\mathcal{DS}_5^{D-2}$, $\mathcal{DS}_5^{D-3}$, and $\mathcal{DS}_5^{D-4}$, while the NCC performance is evaluated with $\mathcal{DS}_5^{D-5}$.

5.2.2.1 Test set

The performances of the CNN using the ICD metric over the test sets of $\mathcal{DS}_5^{D-1}$ are summarized in Table 5.6

Table 5.6: CNN performances expressed with the ICD metric for dataset $\mathcal{DS}_5^{D-1}$.

Metric	$\mathcal{D}$	$\mathcal{H}$
$a[ICD0]$	75.94	68.60
$a[ICD1]$	23.96	31.28
$a[ICD2+]$	0.08	0.12

It is possible to see that roughly 99.92% and 99.88% of the classifications happen in the correct class or within one class from the correct one. Very few cases (four and six, respectively) happen in classes much further away from the correct ones. Moreover, illumination conditions have been observed to be relevant for Didymos prediction with an $ICD1$, while the same cannot be said for Hartley. This demonstrates that the CNN can reduce the search space of the estimated position by grouping almost all cases in or next to the correct class and a considerable number of cases in the correct one.

5.2.2.2 Reduced maps

In this test set, the camera poses and the illumination conditions are the same as $\mathcal{DS}_5^{D-1}$. Three subsets are obtained for each body by selectively removing features from the segmentation maps. This is a useful test to assess their impact on the network's performance and hints at which features are used the most by the networks to predict position classes. In the first subset, both craters and boulders are removed from the segmentation mask ("no features"). In the second subset, only craters are removed ("no craters"), while in the third one, only boulders are removed ("no boulders"). Table 5.7 and Table 5.8 summarize the cumulative accuracies for $\mathcal{D}$ and $\mathcal{H}$.

Table 5.7: CNN performances expressed with the ICD metric for the $\mathcal{DS}_5^{D-2}$ dataset on $\mathcal{D}$.

Metric	No features	No craters	No boulders
$a[ICD0-]$	2.2	66.2	5.1
$a[ICD1-]$	11.8	97.5	21.0
$a[ICD2-]$	20.2	97.8	29.6
$a[ICD3-]$	37.5	98.4	45.7
$a[ICD4-]$	57.3	98.7	60.9
$a[ICD5-]$	72.8	99.0	74.6
$a[ICD6-]$	88.1	99.7	90.5
$a[ICD7-]$	100.0	100	100

Table 5.8: CNN performances expressed with the ICD metric for the $\mathcal{DS}_5^{D-2}$ dataset on $\mathcal{H}$.

Metric	No features	No craters	No boulders
$a[ICD0-]$	60.8	69.1	61.8
$a[ICD1-]$	99.3	99.8	99.4
$a[ICD2-]$	99.5	99.9	99.6
$a[ICD3-]$	99.6	99.9	99.6
$a[ICD4-]$	99.6	99.9	99.7
$a[ICD5-]$	99.6	99.9	99.7
$a[ICD6-]$	99.6	99.9	99.7
$a[ICD7-]$	100.0	100	100

Interestingly, different phenomena are observed for each body. For what concern $\mathcal{D}$, a mildly irregular body, features such as craters and boulders heavily influence the performance. A considerable drop in performance seems primarily caused by the absence of boulders since performance is still high in the "no craters" case but is very low in the "no boulders" case. This hints at the extensive reliance of the network on feature patterns identified in the maps as the primary mechanism to perform position-classification.

On the other hand, for what concern $\mathcal{H}$, a highly irregular body, craters and boulders seem equally important features. Their absence causes a mild drop in performance, hinting at their presence not being the primary mechanism used by the network to perform position-classification.

These interesting results indicate the network being flexible as a global-based and feature-based method. Architectural differences are removed because the network design used for both cases is the same. These results indicate the network's ability to specialize on the outline of $\mathcal{H}$, relying less on surface features when an irregular target body is considered. On the other hand, the network specializes in using surface features whenever the target body presents a regular outline, which could not provide sufficient unique information for position estimation, as in the case of $\mathcal{D}$.

5.2.2.3 Predicted masks

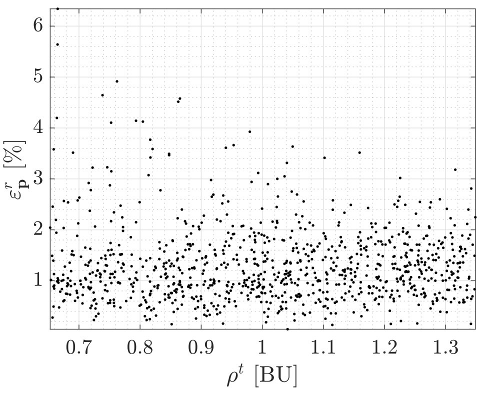
Dataset $\mathcal{DS}_5^{D-3}$ is generated by taking the images associated with the masks of $\mathcal{DS}_5^{D-1}$ and evaluating them over the UNet described in [134]. The purpose of this test is to demonstrate that the CNN can be trained with the true masks and that it can then be deployed for a real application and still work fine with the predicted masks, demonstrating that only a small domain gap exists between the assumed and actual conditions. As it is possible to see from Table 5.9, high values of $a[ICD1-]$ have been retained by using real masks.

Table 5.9: CNN performances expressed with the ICD metric for the $\mathcal{DS}_5^{D-3}$ dataset on $\mathcal{D}$.

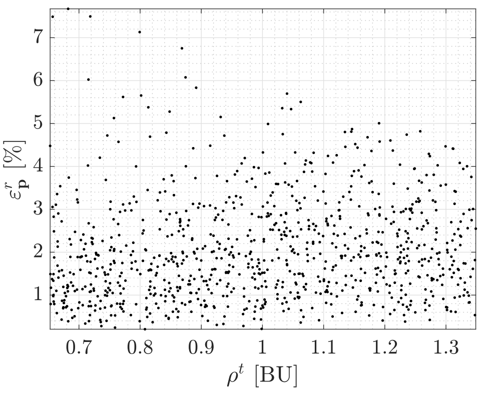
Metric	$\mathcal{D}$	$\mathcal{H}$	Metric	$\mathcal{D}$	$\mathcal{H}$
$a[ICD0]$	52.5	48.1	$a[ICD0-]$	52.5	48.1
$a[ICD1]$	38.3	48.9	$a[ICD1-]$	90.8	97.1
$a[ICD2]$	0.8	1.4	$a[ICD2-]$	91.5	98.5
$a[ICD3]$	1.4	0.3	$a[ICD3-]$	93.0	98.8
$a[ICD4]$	1.4	0.2	$a[ICD4-]$	94.3	98.9
$a[ICD5]$	1.6	0	$a[ICD5-]$	95.9	99.0
$a[ICD6]$	2.1	0.1	$a[ICD6-]$	98.0	99.1
$a[ICD7]$	2.	0.9	$a[ICD7-]$	100	100

5.2.2.4 Normalized Cross-Correlation

Dataset $\mathcal{DS}_5^{D-5}$ is a subset of $\mathcal{DS}_5^{D-1}$ made by 1000 random masks of $\mathcal{D}$ and $\mathcal{H}$ to which the NCC method described in the previous section is applied. The mean positioning error achieved is 63.49 m for $\mathcal{D}$ and 301.06 m for $\mathcal{H}$. In Figure 5.17, it is possible to see the relative percentage error with respect to the true range $\varepsilon_{\mathbf{p}}^r$ as a function of the true range in adimensional space for both bodies. The mean relative percentage error is 1.31% and 2.07% for $\mathcal{D}$ and $\mathcal{H}$. The normalized distance of the two bodies (the distance at which the FOV is saturated by the body) is, respectively, $D_0 = 4.81$ km and $D_0 = 14.41$ km.



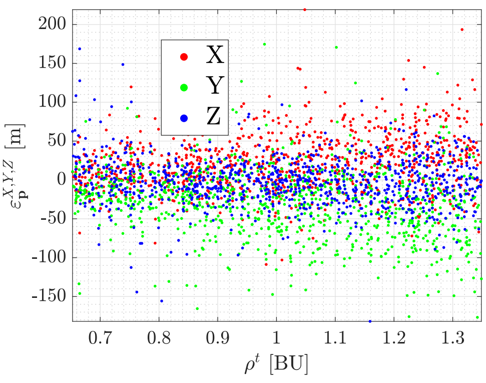
(a) $\mathcal{D}$.



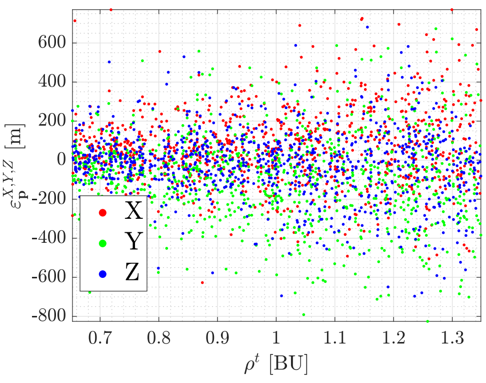
(b) $\mathcal{H}$.

Figure 5.17: Relative percentage error as a function of the range ρ^t from (a) Didymos ($\mathcal{D}$) and (b) Hartley ($\mathcal{H}$). The normalized range D_0 corresponds to $\rho^t = 1$ BU.

In Figure 5.18 and Figure 5.19, the relative positioning percentage error $\varepsilon_{\mathbf{p}}$ is also illustrated in its components in $\mathcal{CAM}$ frame. From these figures, it is possible to observe how, in this case, all components have errors in the same order of magnitude, contributing equally to the total positioning error. Moreover, opposite biases are observed for $\varepsilon^X_{\mathbf{p}}$ and $\varepsilon^Y_{\mathbf{p}}$ in Figure 5.19, which is not observed for the error in the boresight direction.



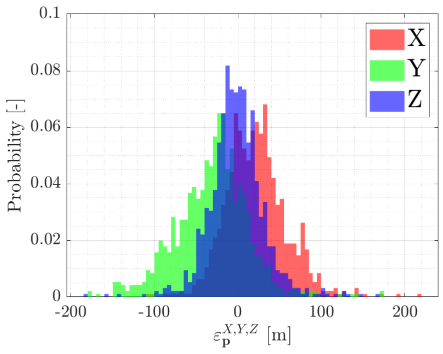
(a) $\varepsilon_{\mathbf{p}}$ by components for $\mathcal{D}$.



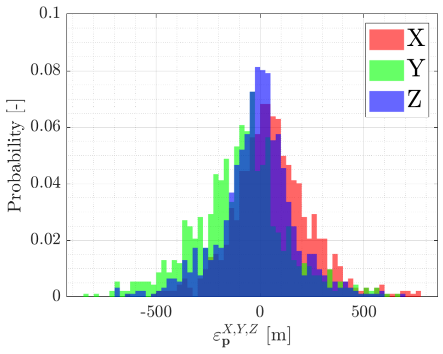
(b) $\varepsilon_{\mathbf{p}}$ by components for $\mathcal{H}$.

Figure 5.18: $\varepsilon_{\mathbf{p}}$ by components in $\mathcal{CAM}$ frame as function of the range ρ^t from (a) Didymos ($\mathcal{D}$) and (b) Hartley ($\mathcal{H}$).

Considering the overall performance, the NCC algorithm proved to work as expected. However, it is reported that the maximum NCC coefficient does not perfectly match with the minimum error location, as it is possible to see from one of such correlation cases in Figure 5.20. This issue may be caused by the absence of the minor boulders in the segmentation mask rendered online and shall be addressed in a real implementation.



(a) Histograms of $\varepsilon_{\mathbf{p}}$ for $\mathcal{D}$.



(b) Histograms of $\varepsilon_{\mathbf{p}}$ for $\mathcal{H}$.

Figure 5.19: Histograms of $\varepsilon_{\mathbf{p}}$ by components in $\mathcal{CAM}$ frame for (a) Didymos ($\mathcal{D}$) and (b) Hartley ($\mathcal{H}$). The binwidth used is of *5* m in (a) and *25* m in (b).

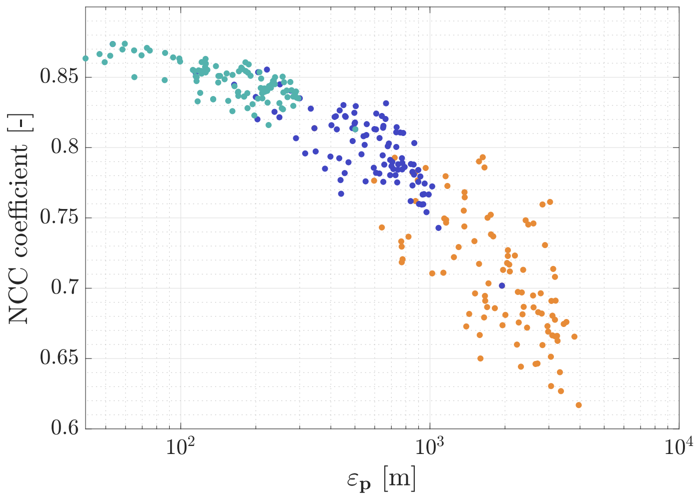


Figure 5.20: Example of the NCC coefficient as a function of the error from iteration 1 (orange), 2 (blue), and 3 (teal).

5.3 Recurrent architectures for navigation

Finally, in this section, the possibility of using image sequences in combination with RNNs is investigated, addressing two major questions: 1) To what extent can short sequences of images improve the position estimate obtained from a single image? and 2) Can RNN perform the same tasks as a Kalman Filter (KF) and be used to generate a position-velocity pair from image sequences?

5.3.1 Training strategy

To obtain the position and/or velocity estimates from a RNN, a training methodology is designed and divided into two main parts, as illustrated in Figure 5.21.

In the first one, CELM architectures are trained to generate position estimates using segmentation maps of Didymos as input. The best architecture is selected and used in inference on a vast test dataset to generate sequences of position estimates, constituting the input for training the RNN. In the second part, an ensemble of RNN is designed and trained to take as input variable sequences of the position estimates previously generated with the best CELM architecture with the goal of either improving the same position estimate or generating a velocity estimate. The sequence of position estimates generated with the CELM can use only optical observables or complement them with rangefinder Light Detection And Ranging (LiDAR) data.

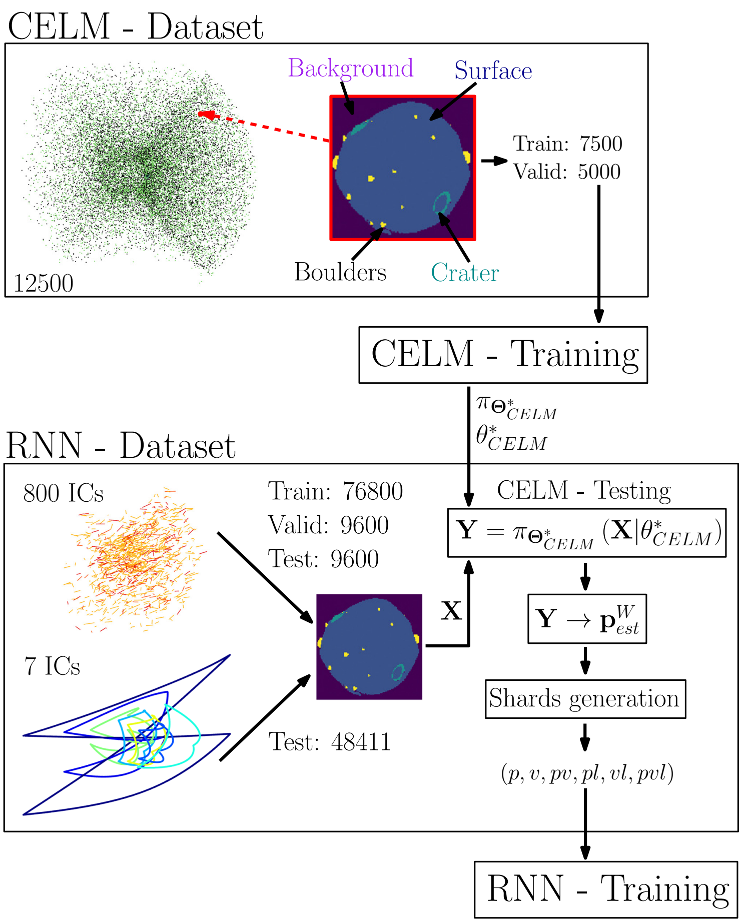


Figure 5.21: Sketch of the combined training strategy for the CELM and RNN architectures.

After a sequence of N position vectors is obtained onboard, the RNN is used to analyze it and produce an estimated state vector valid for the N_{th} instance. This is either made up of only position (p, pl), velocity (v, vl), or both (pv, pvl) components. The l indicates whether or not LiDAR data has been used in generating the position with the CELM. In inference, the CELM and RNN are sequentially applied after a first passage by a UNet that is used to generate the segmentation masks from visible grayscale images, using the procedure illustrated in Section 4.1, schematized in Figure 5.22.

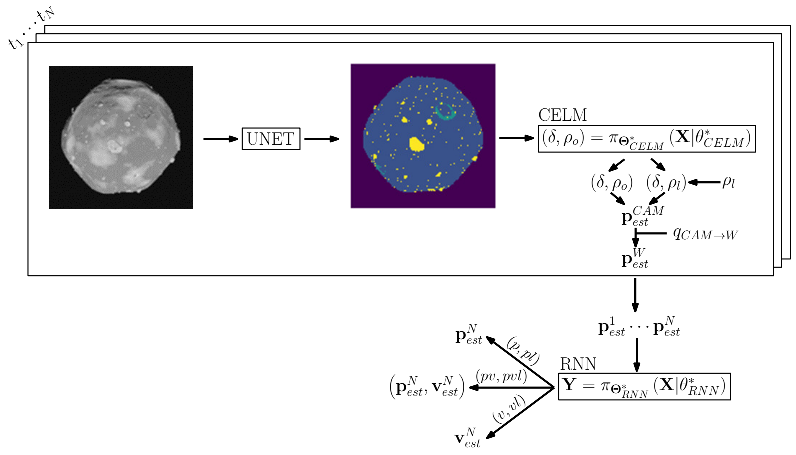

$t_1 \dots t_N$
UNET
CELM
$(\delta, \rho_o) = \pi_{\boldsymbol{\Theta}^*_{CELM}}(\mathbf{X}|\theta^*_{CELM})$
(δ, ρ_o)
(δ, ρ_l)
ρ_l
$\mathbf{p}^{CAM}_{est}$
$q_{CAM \to W}$
$\mathbf{p}^{W}_{est}$
$\mathbf{p}^{1}_{est} \cdots \mathbf{p}^{N}_{est}$
RNN
$\mathbf{Y} = \pi_{\boldsymbol{\Theta}^*_{RNN}}(\mathbf{X}|\theta^*_{RNN})$
(p, pl)
$\mathbf{p}^{N}_{est}$
(pv, pvl)
$(\mathbf{p}^{N}_{est}, \mathbf{v}^{N}_{est})$
(v, vl)
$\mathbf{v}^{N}_{est}$

Figure 5.22: Sketch of the combined architectures in inference.

5.3.1.1 CELM training

Using the $\mathcal{DS}_6$ described in Section A.2.3 made of *12500* segmentation mask and position labels, *600* different convolutional architectures are trained using the CELM paradigm. The training methodology of the CELM is illustrated in the schematic in Figure 5.23.

Each CELM architecture is designed with a hierarchical convolution structure as the one presented in Section 5.1. While going deeper into the network towards the fully connected layer, the starting $128 \times 128 \times 1$ tensor is squeezed; its size is halved while its depth is doubled as a function of the depth level following a power law from 2^4 to 2^8. Each level comprises the consecutive application of convolutions, activation functions, and pooling operations. The convolutions filters are made by 3×3 kernels. Having defined a procedural set of rules to generate each architecture, an ensemble of them is produced with the set of hyperparameters $\boldsymbol{\Theta}$ illustrated in Table 5.10.

Table 5.10: Sets of $\boldsymbol{\Theta}$ explored for the CELM architectures.

Symbol	Possible values
d	1, 2, 3, 4, 5
K_d	Random Uniform (-1,1), Random Normal (0, 1), Orthogonal
A	nReLU, ReLU, tanh, none
P	Mean, Max
C	10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}

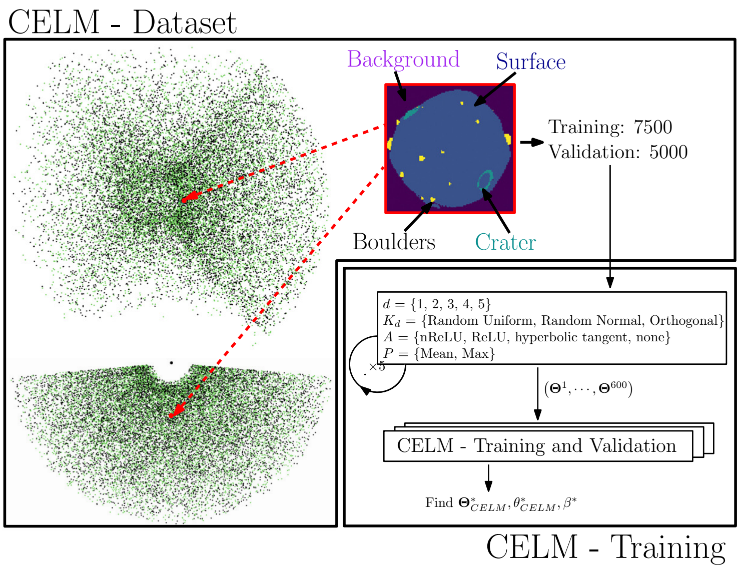

CELM - Dataset
Background
Surface
Training: 7500
Validation: 5000
Boulders
Crater
$d = \{1, 2, 3, 4, 5\}$
$K_d = \{$Random Uniform, Random Normal, Orthogonal$\}$
$A = \{$nReLU, ReLU, hyperbolic tangent, none$\}$
$P = \{$Mean, Max$\}$
$\times 5$
$(\mathbf{\Theta}^1, \cdots, \mathbf{\Theta}^{600})$
CELM - Training and Validation
Find $\mathbf{\Theta}^*_{CELM}, \theta^*_{CELM}, \beta^*$
CELM - Training

Figure 5.23: Sketch of the first part of the training with the CELM. Training and validation split of the $\mathcal{DS}_6$ dataset are represented by the black and green points in the $\mathcal{W}$ reference frame around Didymos. The dataset comprises segmentation maps generated by a camera at these points.

Considering all the possible combinations in Table 5.10, a total of *120* different architectures are generated. To further extend the exploration, the random initialization of each kernel distribution is executed *5* times, producing a total of *600* architectures to train. As in Section 5.1, different values of the regularization term $C$are used with the validation set. The best architecture, $\pi_{\mathbf{\Theta}^*_{CELM}}$, has been initialized using a *Random uniform* distribution of the kernels and is illustrated in Table 5.11.

Using the output of the CELM, the position is generated following the same procedure illustrated in Section 3.1.1.5 and schematized in Figure 5.22, transforming δ, ρ into a LoS vector in the $\mathcal{CAM}$ reference frame and then using assuming to use the attitude quaternion from the star-tracker and the known rigid rotation between the inertial reference frame used by the Star-tracker and the $\mathcal{W}$ frame. As in Section 5.1, no attitude error is simulated in this process. The estimated position in $\mathcal{W}$ frame is thus computed as:

$$\mathbf{p}^{W}_{est} = \mathbf{q}_{\mathcal{CAM}\to\mathcal{W}} \cdot \mathbf{p}^{\mathcal{CAM}}_{est} \tag{5.6}$$

where $\mathbf{p}^{\mathcal{CAM}}_{est}$ is the estimated position in the camera frame and $\mathbf{q}_{\mathcal{CAM}\to\mathcal{W}}$ is the quaternion that rotates from the $\mathcal{CAM}$ to $\mathcal{W}$ frame. In this paper, ρ is either estimated from the images (ρ_o) or with the use of a rangefinder LiDAR sensor (ρ_l). The latter is simulated with the addition of normally distributed noise on the true range ρ_t between CoM of Didymos and the spacecraft as:

Table 5.11: Example of a five layers architecture with a three neurons output layer, made of *392′320* parameters (1.5 MB).

ID	Layer type	Output Shape	Parameters	Connections
I	InputLayer	($\mathcal{B}$, 128, 128, 1)	0	C1
C1	Conv2D	($\mathcal{B}$, 128, 128, 16)	160	A1
A1	nReLU	($\mathcal{B}$, 128, 128, 16)	0	P1
P1	MeanPool2D	($\mathcal{B}$, 64, 64, 16)	0	C2
C2	Conv2D	($\mathcal{B}$, 64, 64, 32)	4640	A2
A2	nReLU	($\mathcal{B}$, 64, 64, 32)	0	P2
P2	MeanPool2D	($\mathcal{B}$, 32, 32, 32)	0	C3
C3	Conv2D	($\mathcal{B}$, 32, 32, 64)	18496	A3
A3	nReLU	($\mathcal{B}$, 32, 32, 64)	0	P3
P3	MeanPool2D	($\mathcal{B}$, 16, 16, 64)	0	C4
C4	Conv2D	($\mathcal{B}$, 16, 16, 128)	73856	A4
A4	nReLU	($\mathcal{B}$, 16, 16, 128)	0	P4
P4	MeanPool2D	($\mathcal{B}$, 8, 8, 128)	0	C5
C5	Conv2D	($\mathcal{B}$, 8, 8, 256)	295168	A5
A5	nReLU	($\mathcal{B}$, 8, 8, 256)	0	P5
P5	MeanPool2D	($\mathcal{B}$, 4, 4, 256)	0	FC
FC	Flattening	($\mathcal{B}$, 4096)	0	O
O	Dense	($\mathcal{B}$, 3)	12291	

$$\rho_l = \rho_t + \sigma_l \cdot \Omega \tag{5.7}$$

where Ω is a normal random distribution function, and σ_l is the LiDAR standard deviation measured as $\sigma_l = \sqrt{\sigma_h^2 + \sigma_s^2}$, σ_h being the contribution by the instrument uncertainty (assumed to be *1* m as an educated guess from the DLEM rangefinder LiDARs from Jenoptik [33]) and σ_s the uncertainty provided by the deviation of the small-body shape ($\sigma_s = 13.63$ m) due to irregularities from a sphere centered in the CoM with a radius equal to the mean value of *392.48* m.

Note that the current modeling of the LiDAR simplifies the observation acquisition by only introducing noise in the measurement while disregarding other significant effects. In a real LiDAR, other sources of error like range-dependent noises, pointing contributions, shape uncertainty effects, and correlation between different sources of errors would further degrade LiDAR performances. A simplified modeling of the LiDAR is therefore adopted for what concerns the analysis presented in this section.

5.3.1.2 RNN training

The RNN is designed after a thorough hyperparameters search using a combination of LSTM cells and a single layer of neurons. The RNN takes as input a sequence of N previously estimated positions obtained with the best CELM identified in the

[33] https://www.jenoptik.com/products/lidar-sensors-technologies/laser-rangefinders/oem-modules-system-integration/dlem, last accessed 27th of October 2022.

first part of the training and produces the current estimate of the position, velocity, or both, with and without the use of LiDAR. The entire training procedure of the RNN is illustrated in Figure 5.24.

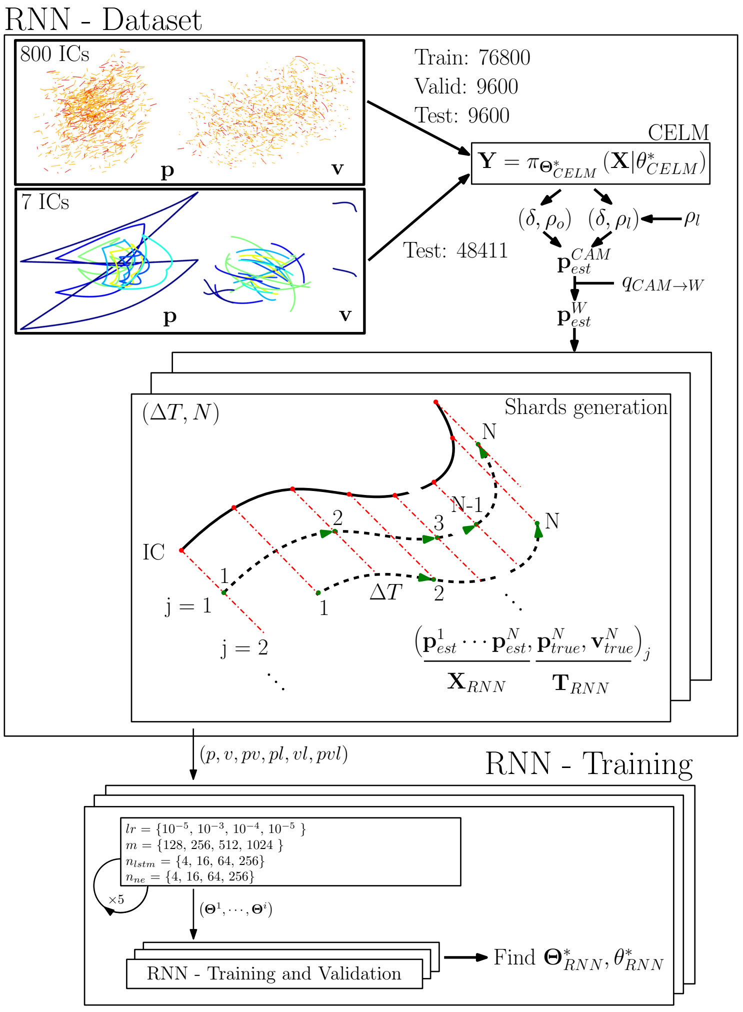

RNN - Dataset
800 ICs
p
v
7 ICs
p
v
Train: 76800
Valid: 9600
Test: 9600
Test: 48411
CELM
$\mathbf{Y} = \pi_{\Theta^*_{CELM}}(\mathbf{X}|\theta^*_{CELM})$
(δ, ρ_o)
(δ, ρ_l)
ρ_l
$\mathbf{p}^{CAM}_{est}$
$q_{CAM \to W}$
$\mathbf{p}^{W}_{est}$
$(\Delta T, N)$
Shards generation
IC
j = 1
j = 2
1
2
3
N-1
N
ΔT
$\left(\mathbf{p}^1_{est} \cdots \mathbf{p}^N_{est}, \mathbf{p}^N_{true}, \mathbf{v}^N_{true}\right)_j$
$\mathbf{X}_{RNN}$
$\mathbf{T}_{RNN}$
(p, v, pv, pl, vl, pvl)
RNN - Training
$lr = \{10^{-5}, 10^{-3}, 10^{-4}, 10^{-5}\}$
$m = \{128, 256, 512, 1024\}$
$n_{lstm} = \{4, 16, 64, 256\}$
$n_{ne} = \{4, 16, 64, 256\}$
×5
$(\Theta^1, \cdots, \Theta^i)$
RNN - Training and Validation
Find $\Theta^*_{RNN}, \theta^*_{RNN}$

Figure 5.24: Sketch of the second portion of the training with the RNN.

Using the train and validation shards of $\mathcal{DS}^R_6$ with the properties illustrated in Table A.17 in combination with the six different labeling strategies (p, v, pv, pl, vl, pvl), an extensive hyperparameter search for the definition of the best RNN architectures is performed.

Each RNN is generated in TF using a number of LSTM cells driven by n_{lstm} parameter. As an activation function of the cells, a *hyperbolic tangent* is used while a *sigmoid* is used as a recurrent activation function. The LSTM cells are followed up by a single layer made up of n_{ne} neurons, which uses the *ReLU* as an activation function. The hyperparameter search serves the purpose of identifying the values of n_{lstm}, n_{ne} , as well as m and lr , that characterize the best performing RNN architecture on the validation set. All the values of hyperparameters tested during training of the RNN are illustrated in Table 5.12 while a generic RNN architecture is illustrated in Table 5.13.

Table 5.12: Sets of $\boldsymbol{\Theta}$ explored for RNN training.

Symbol	**Description**	**Possible values**
lr	Learning rate of *Adam* optimizer	10^{-3},10^{-4}, 10^{-5}
m	Batch size for gradient descent	128, 256, 512, 1024
n_{lstm}	Number of LSTM cells used in the RNN	4, 16,64, 256
n_{ne}	Number of neurons used in the RNN architecture	4, 16, 64, 256

Table 5.13: Example of a generic RNN architectures. The number of parameters is omitted as it varies with n_{lstm} and n_{ne}.

ID	Layer type	Output Shape	Connections
I	InputLayer	($\mathcal{B}$, 5, 3)	L
L	LSTM	($\mathcal{B}$, n_{lstm}, 6)	D1
D1	Dense	($\mathcal{B}$, n_{ne}, 6)	R1
R1	ReLU	($\mathcal{B}$, n_{ne}, 6)	D2
D2	Dense	($\mathcal{B}$, 6)	O
O	ReLU	($\mathcal{B}$, 6)	

Adopting the methodology illustrated in [144] , recommending an iterative framework with incremental exploration of the hyperparameter space, with the empirical findings from [76], stressing the importance of gradual changes and global perspective while performing hyperparameters tuning, a *5* steps iterative grid-search is performed using *adam* optimizer with default settings and MSE as loss function. The number of parameters is reduced at each step, while epochs are increased to keep a consistent computational effort throughout different steps. A tournament training is thus performed using the best mean position and/or velocity errors as global metrics, using the $\varepsilon_{\mathbf{p}}^{r}$ and $\varepsilon_{\mathbf{v}}^{r}$ metrics defined in Section 1.8.5.

The best-performing architectures are obtained by repeating this methodology for each of the six labeling strategies considered for the RNN. Their characteristics are summarized in Table 5.14.

Table 5.14: Best hyperparameters $\boldsymbol{\Theta}^*$ of the RNN for each labeling strategy.

Label	**p**	**v**	**p, v**	**p**	**v**	**p, v**
LiDAR	No	No	No	Yes	Yes	Yes
Name	R_p	R_v	R_{pv}	R_{pl}	R_{vl}	R_{pvl}
ΔT [s]	150	3600	3600	150	3600	3600
N	30	5	5	30	5	5
lr	10^{-2}	10^{-3}	$2 \cdot 10^{-4}$	10^{-3}	10^{-3}	10^{-3}
m	128	256	128	64	256	64
n_{lstm}	16	128	128	256	512	256
n_{ne}	16	64	512	32	512	256

5.3.2 Results

In this section, the position and velocity onboard reconstruction results are illustrated for the test sets of $\mathcal{DS}_6$. For a better interpretation, a consistent color scheme and notation is adopted. The performance of the RNNs is characterized by a palette of blue colors of the *jet* colormap while the CELMs ones by red colors of the same colormap, as it is possible to see from the legends in Figure 5.3 and Figure 5.27. Also, the performance related to the CELM are annotated as C_i, the ones related to the RNN as R_i, and the i representing the labeling strategy. The metrics used to assess the position and velocity reconstruction are Equation 1.20 and Equation 1.21 defined in Section 1.8.5.

In Table 5.15, it is possible to see a summary of the performance of the various methods with different labeling strategies on two different test sets. Te_1 represents random short trajectories designed around the Didymos asteroid, while Te_2 represents specific closed orbits about Didymos referred to as Close Proximity Orbit (CPO)s, as illustrated in Section A.2.3. The coupling between LiDAR data and optical observables is beneficial in terms of performance, which is one order of magnitude better than estimates generated with optical observables alone. It is also noted that the RNNs reconstruct a better estimate whenever they are solely focused on the position or velocity, as in the p, v, pl, and vl cases, and not in the mixed labeling strategies such as in the pv and pvl cases.

In Figure 5.25, the distributions of $\varepsilon^r_{\mathbf{p}}$ and $\varepsilon^r_{\mathbf{v}}$ are illustrated with box plots for the various labeling strategies on both test sets (Te$_1$ first and Te$_2$ second). For completeness, in Figure 5.27, it is possible to see the same data generating cumulative performance plots (Te$_1$ solid and Te$_2$ dashed) as well as histogram plots in Figure 5.26.

From these figures, it is possible to appreciate better the same trends identified by the global metrics in Table 5.15. In particular, the order of magnitude improvement in the performance when considering the LiDAR, the capability of the RNN to reconstruct the velocity, especially with data from the LiDAR, and finally the capability of the RNN to improve the position estimate, albeit only marginally,

Table 5.15: Performance of the state reconstruction by CELM (C) and RNN (R) with different labeling strategies. The values expressed are the $\mu(\varepsilon^r_{\mathbf{p}})$ or $\mu(\varepsilon^r_{\mathbf{v}})$ and their corresponding (σ).

Labels	Dataset	C	R	R
p	Te*1*	2.69 (2.15)	1.28 (1.25)	-
	Te*2*	2.70 (2.01)	1.30 (0.87)	-
v	Te*1*	-	-	31.86 (27.75)
	Te*2*	-	-	33.67 (29.29)
pv	Te*1*	2.67 (2.05)	2.74 (1.66)	34.46 (28.82)
	Te*2*	2.69 (2.00)	2.74 (1.67)	41.16 (33.12)
pl	Te*1*	0.24 (0.21)	0.22 (0.44)	-
	Te*2*	0.26 (0.18)	0.18 (0.12)	-
vl	Te*1*	-	-	4.07 (3.54)
	Te*2*	-	-	5.57 (7.71)
pvl	Te*1*	0.23 (0.20)	0.90 (0.68)	4.72 (4.71)
	Te*2*	0.26 (0.18)	0.87 (1.66)	6.25 (7.90)

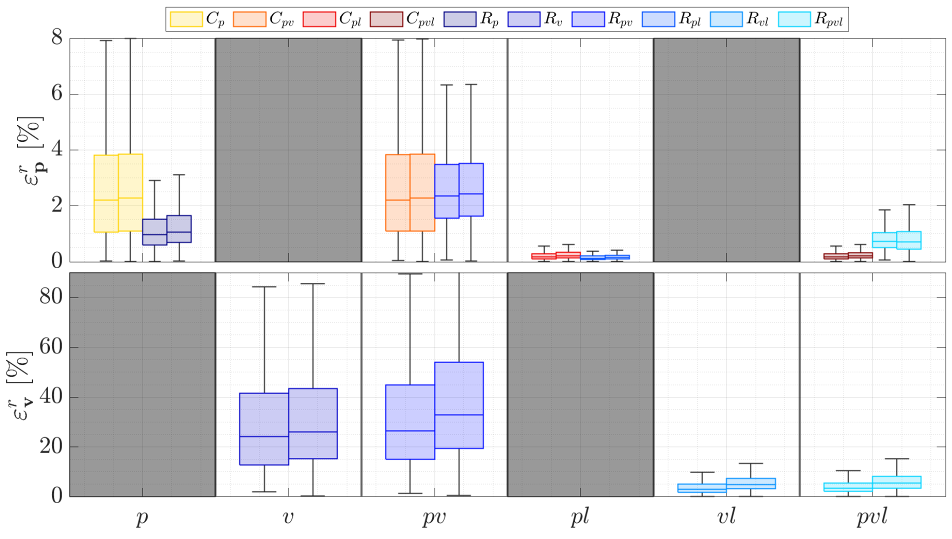


Figure 5.25: Box plot of the position (top) and velocity (bottom) reconstruction error.

as it is possible to observe by comparing C_p or C_{pl} with R_p or R_{pl}. Finally, it is also commented that the training and validation envelopes chosen for the RNN demonstrated to have been chosen adequately for the testing conditions.

Finally, it is interesting to observe the reconstructed states of the CPOs of the Te$_2$ in the $\mathcal{W}$ reference frame both in the position and velocity phase spaces respectively in the scatter plots in Figure 5.28 and Figure 5.29. First, an oscillation

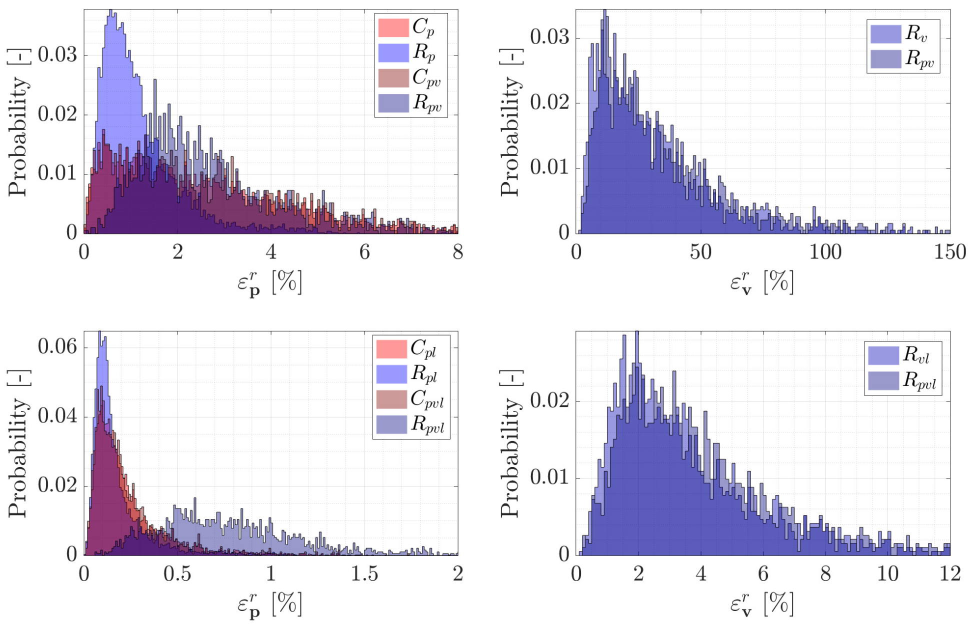


Figure 5.26: Histogram performance of the position (left) and velocity (right) reconstruction error. The binwidth from top to bottom, left to right, are *0.05*, *1*, *0.01*, and *0.1*.

compatible with the rotational state of Didymos is observed to impact the estimated position in the radial direction in Figure 5.28. This seems to be caused by position estimates by C_p, also reflected in R_p. These oscillations are visible in the close-up view of the CPO trajectories projected in the XY plane of the $\mathcal{W}$ reference frame in Figure 5.28. Second, using LiDAR data, as in the R_{pl} and C_{pl}, generates estimates much closer to the true CPOs. The oscillation phenomenon is still present but limited in amplitude due to the LiDAR capability to generate a better estimate.

Generating the same visualization in the velocity phase space, Figure 5.29 is obtained. It is possible to see that the estimates without the LiDAR (R_v) are much more loosely related to the true CPOs in velocity phase space, while the ones obtained with the LiDAR (R_{vl}) are indeed much more adherent to the true ones.

Additionally, two interesting phenomena are observed. First, by comparing the training interval in Figure A.21 with the testing one of the various CPOs, it is possible to see that CPO_1 is the only one that is not entirely contained within the training envelope. In particular, two regions, with positive and negative values of V_y, are observed outside the training envelope. This has been a deliberate design choice to test RNN generalization capabilities. Second, of the two groups of trajectories in the velocity phase space of CPO_1 divided by the sign of V_y, it is possible to see that only the ones with positive V_y get reconstructed well in all three components. Interestingly, the ones with negative V_y are reconstructed well only in the X and Z components, as visible from the XY view in Figure 5.29.

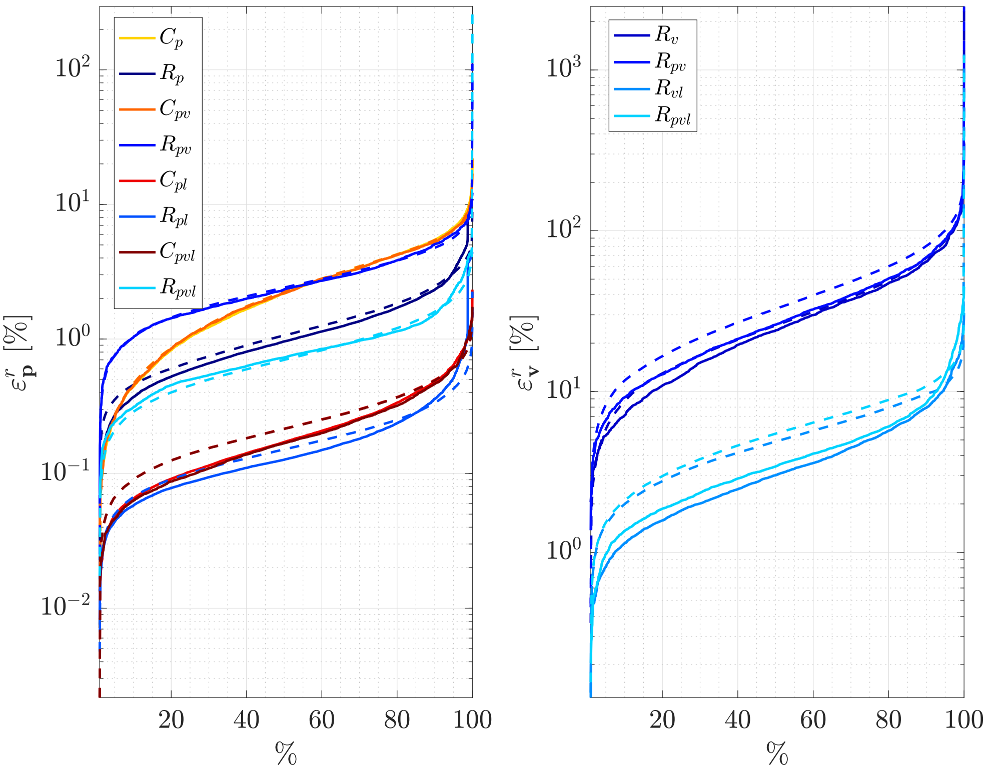


Figure 5.27: Cumulative performance of the position (left) and velocity (right) reconstruction error.

From these results, a twofold conclusion is drawn. First, RNN architectures can be used to improve the reconstruction of the position vector using a sequence of images. This improvement, however, is negligible, especially considering the case in which LiDAR data is used. Second, regarding the velocity, the RNN cannot generate an accurate estimate using only optical observables, which prompts the use of LiDAR data to decrease the positioning error in the radial direction. When optical observables are complemented by LiDAR data, the RNN can accurately reconstruct the velocity. This is partially true outside the training envelope, showing promising generalization capabilities. Finally, non-mixed predictions were simpler and more accurate when considering a mixed labeling strategy in which the position and velocity are estimated by the RNN.

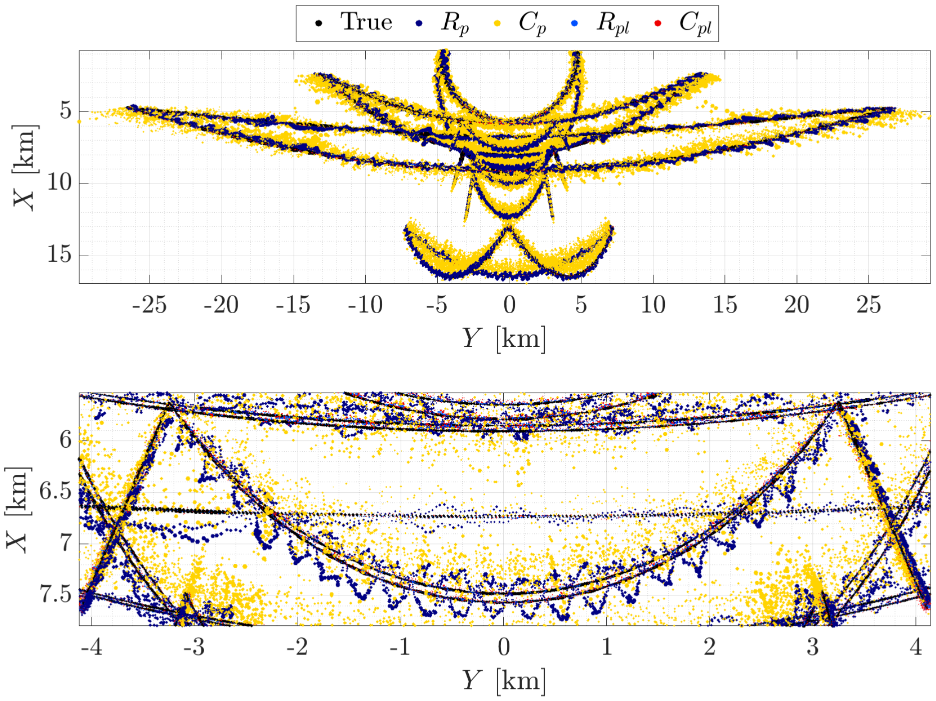


Figure 5.28: True and estimated positions by C_p, C_{pl} and R_p, R_{pl}. The size of the points is proportional to $\varepsilon^r_{\mathbf{p}}$.

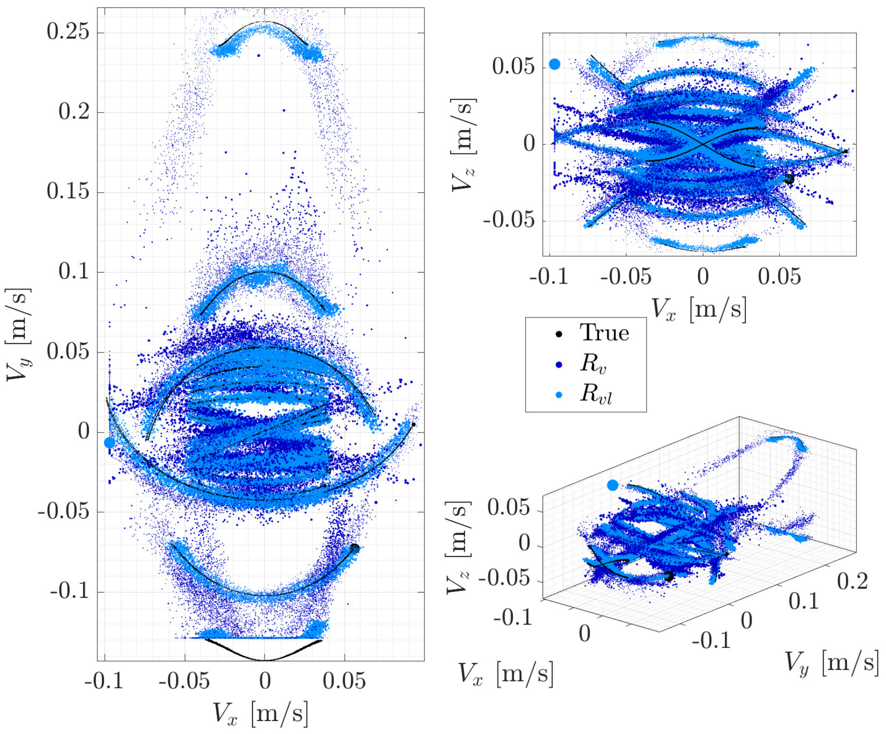


Figure 5.29: True and estimated velocities by R_v and R_{vl}. The size of the points is proportional to $\varepsilon_{\mathbf{v}}^r$.

5.4 Final remarks

In this chapter, three navigation approaches around small bodies that make use of AI techniques are investigated in detail. In particular, navigation is addressed using regression, classification, segmentation, and methods, considering as input both grayscale images or segmentation maps, exploring end-to-end "from image to position" and intermediate "from image to observables" pipelines with convolutional, correlation, and recurrent approaches.

Particular emphasis has been put on the importance of architectural choices, complex bootstrap training methodologies, labeling strategy, and reference frames used. What follows is a list of final remarks.

- An extensive analysis is performed for convolutional architectures, investigating the impact of three different reference frames, five different labeling strategies, four different irregular bodies, and four architectures with the purpose of:
  - Investigating whether simple convolutional architectures trained with the CELM paradigm can perform similarly to CNNs regarding space images of small bodies in the medium regime.
  - Developing a methodology that uses CELM fast training time to explore the architecture design space of CNNs
  - Determining the best labeling strategy and reference frames for position estimation tasks with convolutional architectures.
- The analysis resulted in tens of thousands of different architectures being explored with various training instances and concluded that:
  - CELM-trained networks can be considered a valid alternative only when using the (δ, ρ) labeling strategy. Their performance are similar to those of CNN, especially when regular shape bodies are considered. Overall, CNNs outperforms all architectures considered.
  - CELM can be successfully and efficiently used to explore the architecture design space of CNNs. Combining exploration with a first pass to find the best possible architectures and exploitation with a second one dedicated to optimizing their weights and biases guarantees efficient and robust network development. This training framework can be adapted for a real mission case to use the resources available for training and reduce costs efficiently.
  - It is found that the coupling between IP method and labeling strategy plays a fundamental role. In particular, the (δ, ρ) labeling strategy is the simplest and most efficient form of label representation for position estimation. This is attributed to a direct link between estimated labels and input images. The positioning error with such a strategy suffers the highest in the radial direction. Regarding the other ones considered, no significant difference is observed between polar and cartesian coordinates. At the same time, the choice of the reference frame plays a considerable role, the $\mathcal{W}$ being preferred over $\mathcal{AS}$. This is attributed to the influence

of illumination conditions over the global shape of the body.

- A classification-correlation architecture is developed using segmentation maps as input. The approach is advantaged by using simple, low-resolution models for online rendering and the correlation not being affected by local pixel-intensity variations due to illumination conditions and albedo modeling. The architecture is performed with a three parts structure: A UNet converts grayscale images to segmentation maps, which are used by a CNN to perform classification to identify the rough position of the spacecraft around the body in the $\mathcal{W}$ reference frame. Lastly, this is refined with an iterative correlation scheme based on a NCC metric.
- The classification CNN demonstrated its capability to generate robust, rough predictions in or next to the correct classes, with illumination conditions considered relevant in the Didymos case but not in the Hartley one.
- Features such as craters and boulders were pivotal for the network trained on Didymos, a mildly irregular body, hinting at its features-based disposition developed during training. Contrarily, the global outline of Hartley, an irregular body, turned out to be the most important feature used by the network in predicting a class, hinting at its global-based disposition. Both results indicate that the network is flexible enough to specialize with patterns of local features and the shape outline, depending on the properties of the target body.
- The NCC-based algorithm performs well in refining the position estimate around the target bodies, with a minor issue caused by the maximum NCC metric not scoring close to the location of the minimum position error.
- A set of RNN architectures is designed and trained with various sequences of position estimates obtained from optical observables of segmentation maps extracted by CELM, optionally complemented with LiDAR data in the radial direction. This activity has been motivated by the interest in investigating:
  - To what extent can short sequences of images improve the position estimate obtained from traditional convolutional approaches applied to single images?
  - Can RNN perform the same tasks as a KF and be used to generate a position-velocity pair from image sequences?
- Investigating different combinations of labeling strategies, mixed use of LiDAR data in the boresight direction, and assessing the performance within and outside the nominal training envelope, the following findings have been found:
  - Using a sequence of position estimates, a RNN improvement is marginal in increasing the accuracy of the position estimate.
  - The coupling between LiDAR data and optical observables is beneficial in terms of performance, which is one order of magnitude better than estimates generated with optical observables alone.
  - A RNN can successfully generate accurate velocity estimates from position sequences, but only with the use of LiDAR data. This is also partially true outside the training envelope, showing promising

generalization capabilities.
  - In all RNN architectures considered, a single three-value position or velocity vector estimate has proven to be more accurate than a combined six-value position and velocity vector.
  - An oscillation compatible with the rotational state of Didymos is observed to impact the estimated position in the radial direction.
- The RNN architectures do not have an a-priori initialization of the state, which could be relevant for autonomous operations. On the other hand, the environment dynamic is embedded in the data used for training of the RNN. While an update of the dynamic in a KF could be implemented with a simple parameter change or by a different implementation of a given dynamical model, in the case of the RNN it could require new training.
- Different dynamical settings could be used to verify the generalization capability of the RNN.
- The LiDAR is assumed to be available irrespective of the range from the asteroid, which is an assumption that may be invalid depending on the specific hardware considered. Including the simulation of a duty cycle of the LiDAR, which could be activated only below a predetermined distance, would be interesting.
- The LiDAR is represented with a too-optimistic modeling. The assumptions adopted in this chapter could be revisited to increase the model's fidelity. Additional noise sources and a different modeling strategy could be adopted to represent the sensor observation more realistically.
- It is noted that in both centroiding-based and spherical coordinates strategies illustrated in Section 5.1 and Section 5.3, most of the error is generated along the radial direction. On the other hand, the convolutional network in Section 5.1 based on cartesian coordinates and the correlation strategy illustrated in Section 5.2, distributes the error across all axes.
- Compared to traditional approaches, AI-based solutions have been demonstrated to improve both centroiding and positioning performance with respect to onboard implementations.
- Considering positioning performance, Section 5.1 illustrates how convolution architectures, with the proper labeling strategy, can achieve relative positioning errors $\varepsilon^r_{\mathbf{p}}$ between 2% and 3% for $\mathcal{D}$, $\mathcal{H}$, $\mathcal{L}$, and $\mathcal{P}$, as illustrated in Figure 5.3 and Figure 5.4. These values are solely generated by the IP, before the eventual application of filtering techniques or recurrent architectures. The latter are investigated in Section 5.3 for $\mathcal{D}$, demonstrating to improve the relative positioning error from 2.7% to 1.3% or from 0.25% to 0.20% when complementing optical observables with a LiDAR in the boresight direction. These onboard position reconstruction performance can be compared with other traditional IP approaches presented in Section 6, such as those based on IP + filtering, which use the COB and SSWCOB IP algorithms. For example, comparing the performance illustrated in Figure 6.28, conveniently represented with a 1% error line, it is possible to appreciate the higher

performance of the AI-enhanced methods whenever the COB method is used. On the other hand, similar results are achieved only by combining the IP output of the SSWCOB with filtering capabilities.

- Navigating using segmentation maps, classification networks, and correlation schemes is investigated for $\mathcal{D}$ and $\mathcal{H}$ in Section 5.2. The classification framework provides only a rough position (as it will be illustrated in Figure 6.42(a), with an error of 9.92%) which is further refined by one order of magnitude by the normalized cross-correlation scheme, passing to $\varepsilon^r_{\mathbf{p}}$ values of 1.31% for $\mathcal{D}$ and 2.07% for $\mathcal{H}$. The navigation performance based on the correlation method with segmentation masks is only marginally better than that of the best AI-centroid methods. Comparing these results with Stereo Photo Clinometry (SPC), the performance with respect to ground-based applications is unmatched, as SPC is capable of generating cm-level position reconstruction errors [146], which are about two to three orders of magnitude lower compared to the few percentage error reported in this work. However, a better performance is reported considering onboard SPC implementations, such as the one in [147], which achieve in the best and worst case scenarios values of relative positioning error between 8.7% and 24%.
- Considering the centroiding performance in the image plane, AI-based centroiding estimates are generated with an error between 1 px and 2 px, as reported in Figure 5.7 for 1024×1024 images. A comparison of this performance with a traditional one is illustrated by the error ellipses in Figure 5.8 with the unscattered CoB case, given that from the analysis in [30] (Figure 16 and Figure 17 of [30]) and [35], the CoB can be considered a traditional and robust center-finding algorithm for irregular bodies. Further comparing these error ellipses with those presented in Figure 6.36(c), demonstrates the jump in accuracy whenever convolutional architectures are used instead of unscattered IP methods, data-driven scattering ones, and NN architectures.
- The results obtained are also compared with other external works. For example, both the Lambertian sphere correlation algorithm presented in [40] (see Figure 12 of [40]) and the scattering CoB function in [27] (See Figure 5, Figure 6, and Table 3 of [27]), exhibits errors in the order of tens of pixels. This is comparable with the performance of the WCOB and NN approaches (See Figure 6.36), which have an error one order of magnitude higher compared to CNN approaches. Note that both [27, 40] assume 1024×1024 images and trajectories that are comparable with those analyzed in Section 5.1 and Section 6.4.1.
- Finally, the results presented in this section seem coherent with the ones presented in other relevant works adopting convolutional architectures around small bodies for centroiding and position reconstruction [43, 44].

6 Milani

"The dinosaurs became extinct because they didn't have a space program."

Larry Niven

Of all the population of minor bodies, NEA are characterized by orbital parameters close to those of Earth, making them accessible targets even with low-cost and small platform missions. CubeSats, which are modular miniaturized spacecraft of several units (1 unit being a box of side *10* cm), are revolutionizing the way Solar System exploration is made by diversifying and complementing the scientific objectives of larger missions [1, 2, 7]. CubeSats can be exploited as opportunistic payloads to be deployed in situ once the main spacecraft has reached its target. This approach is adopted in the Asteroid Impact and Deflection Assessment (AIDA) collaboration between the NASA and the ESA to study and characterize an impact with the Didymos asteroid system [148].

As part of this collaboration, NASA launched the Double Asteroid Redirection Test (DART) kinetic impactor spacecraft [19, 149], whose impact with the secondary asteroid of 65803 Didymos has been observed by LICIACube in 2022 [150]. In October 2024, ESA will launch the Hera mission to investigate the dynamical and geological properties of the binary system [114, 151]. Hera will release two 6U deep-space CubeSats, named Juventas [152] and Milani [153, 154], to map and characterize the system. They will be the first interplanetary CubeSats to execute long-term operations in the proximity of a binary asteroid system. The nominal duration of both missions is set to 90 days, with a backup option for a further 90-day extension.

This chapter details the work performed on the architecture design and performance analysis of the semi-autonomous vision-based GNC subsystem of Milani in the two main operative phases of the mission.

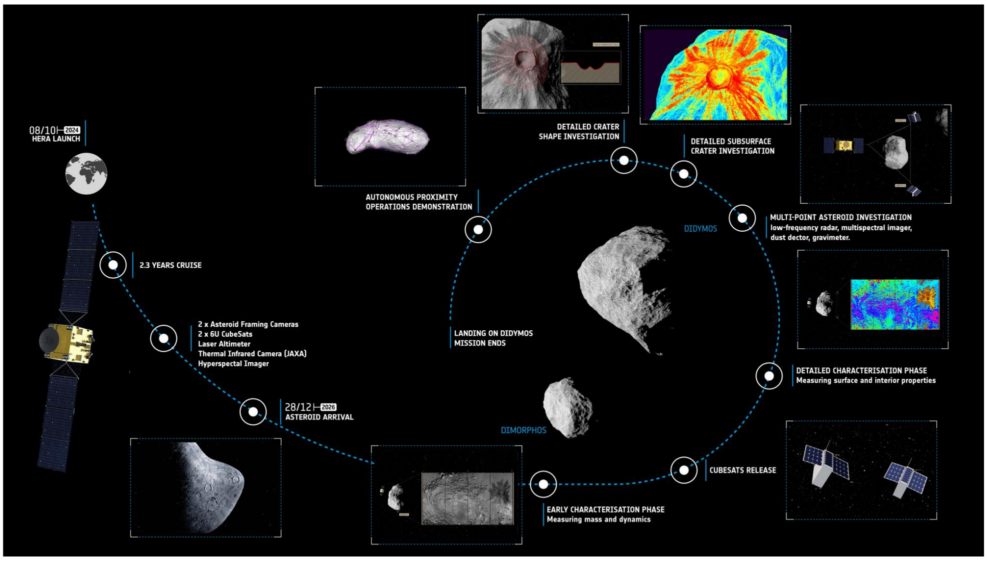


Figure 6.1: High-level overview of the Hera mission. Credits: ESA.

6.1 Mission overview

Milani is a 6U CubeSat that the Hera mothercraft will release in the Didymos environment in early 2027 after an earlier characterization of the binary system. The Didymos system consists of a primary and a secondary, respectively Didymos and Dimorphos, also called Didymos (D1) and Dimorphos (D2) in the rest of the chapter. The former is estimated to be an irregular, spherical-like body with a diameter of 780 m, while the latter is currently modeled as a tri-axial ellipsoid with a major axis of 170 m.

The main scientific and technological objectives of the Milani mission are to:

Characterize the Didymos binary system. This includes supporting Hera in determining the system's extrinsic properties, characterization of the asteroids' surfaces, evaluation of space weathering phenomena, and characterization of the crater created by DART. This objective will be achieved by a global mapping of D1 and D2 with high-resolution images of both bodies with the ASPECT payload [155].

Estimate the gravity field. The range and range-rate measurements exchanged between Milani and Hera via the Inter-Satellite Link (ISL) are exploited to estimate the gravity field in the asteroid environment.

Characterize the dust environment. This includes detecting inorganic materials, volatiles, and light organics within the asteroid environment and deep space. The VISTA sensor will fulfill this objective [156].

Demonstrate ISL communication with Hera. This targets the capability of communicating with a data-relay spacecraft for payload and platform data transmission in deep space with a CubeSat.

Demonstrate the use of CubeSat technologies in deep space. This includes the capability of flying a CubeSat in an asteroid environment, determining the position with vision-based methods, and showing the use of miniaturized technologies in a harsh environment.

To accomplish these objectives, Milani is designed with orbital and attitude control capabilities. The platform is a 6U CubeSat with deployable solar arrays. In addition to ASPECT, VISTA, and the ISL antennas, Milani will be equipped with a wide FOV navcam, a LiDAR, two Sun Sensor (SS), a Star Tracker (STR), an Inertial Measurement Unit (IMU), thrusters and a set of Reaction Wheel (RW).

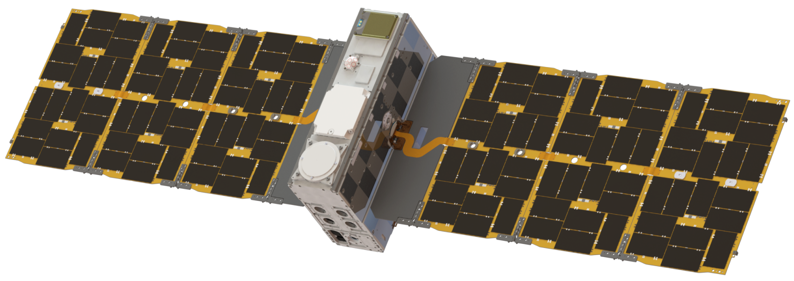


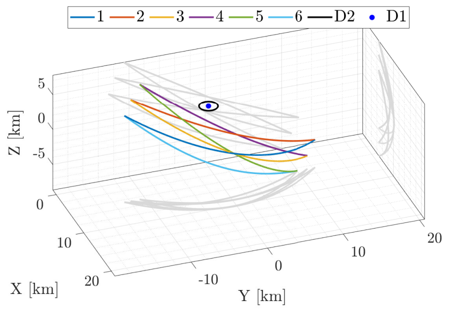
Figure 6.2: Artistic representation of the Milani CubeSat. Credits: Tyvak International.

The mission consists of several phases, which are: 1) ejection from the mothercraft and commissioning; 2) transfer to the operational phase, where the CubeSat is accompanied to achieve operational orbit; 3) the Far Range Phase (FRP), where the system is characterized from large distances; 4) the Close Range Phase (CRP), where the system is characterized from closer distances with high-risk flyby arcs; 5) the EXperimental Phase (EXP) where the CubeSat will orbit on a Sun-Stabilized Terminator Orbit (SSTO); 6) a decommissioning phase where Milani will either be injected into a heliocentric graveyard orbit or will attempt a soft landing on D2.

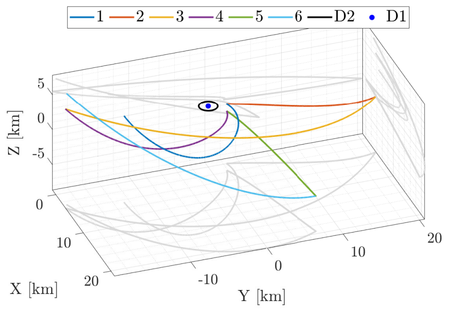
The two main operative phases of the mission are the FRP and CRP, which are strongly influenced by the effect of the binary system gravity and Solar Radiation Pressure (SRP) perturbation [153, 154]. Milani's trajectories in these two phases develop as passively safe hyperbolic arcs, represented in Figure 6.3. The trajectories are illustrated in the convenient $\mathcal{W}$ reference frame, which is centered on D1, with the Z-axis aligned with its spin axis, and the X and Y-axes co-planar with the orbital plane of D2, with the X-axis following the projection of the Sun in such plane.

The FRP and CRP last 21 and 35 days, respectively. The FRP alternates 3 and 4-day arcs, while in the CRP there are also 7-day arcs with corrective maneuvers in the middle, as the CubeSat gets significantly closer to the system than in the FRP. The portions of an arc in the CRP before and after the correction maneuver are referred to as *a* and *b*, respectively. The FRP exhibits symmetrical arcs that develop within 9-14 km from D1, while the CRP is constituted by asymmetrical arcs with a range of 3-22 km from the system.

Note that for simplicity, the same color code associated with each arc of the FRP and CRP illustrated in Figure 6.3 will be used across the chapter to represent properties and performance of various algorithms on specific arcs.



(a) Arcs of the FRP.



(b) Arcs of the CRP.

Figure 6.3: Milani's arcs in FRP and CRP. Gray lines represent projections into the X-Y, X-Z, and Y-Z planes.

Milani's GNC subsystem is designed as a semi-autonomous vision-based system with the primary purpose of generating a reliable, simple, and accurate primary pointing to provide to the Attitude Determination and Control System (ADCS) during the different scenarios of the mission. To do so, Milani's GNC exploits strategies based on IP algorithms extracting optical observables from images of the binary system.

The GNC and ADCS are two separate but deeply interconnected subsystems. Together, they form the Attitude and Orbital Control System (AOCS), which is responsible for the full six-degree-of-freedom orbital and attitude navigation, guidance, and control of the CubeSat. Since the GNC system generates autonomously an onboard primary pointing profile as output for the rotational motion, and since it does not have onboard autonomy on the translational guidance and control, the system is defined as semi-autonomous. The overall architecture of the AOCS is represented in Figure 6.4, where the connections between the GNC, the ADCS and the rest of the system are visualized.

Both the GNC and IP have been developed by the DART[34] at Politecnico di Milano using Simulink 2020a[35] for its simplicity and the capability to convert high-level rapid prototyping code in Matlab/Simulink as C-code via auto-coding. This proved fundamental for fast iterations between the design of the algorithms and their integration with the onboard software. The IP software is set to run

[34]https://dart.polimi.it/, last accessed: 27th June 2022.

[35]https://www.mathworks.com/products/simulink.html, last accessed: 27th June 2022.

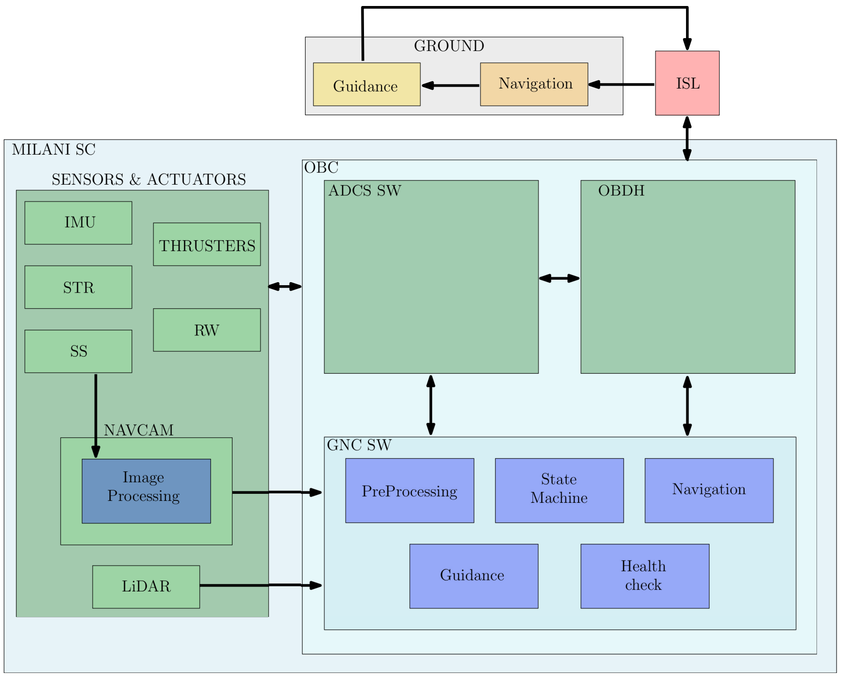


Figure 6.4: High-level architecture of the AOCS system of Milani. The areas highlighted in blue are the focus of this chapter.

on-demand within Milani's NavCam DM3730 processor whenever a new image is available. Currently, the IP is designed in Simulink[36] to be easily interfaced with the rest of the GNC. Exploiting the auto-coding capabilities of Simulink, the IP code is translated into C and deployed as onboard software.

The Milani consortium is composed of entities and institutions from Italy, Czech Republic, and Finland. The consortium prime is Tyvak International, which is responsible for the whole program management and platform design, development, integration, testing, and final delivery to the customer. Politecnico di Torino has worked on the requirements definition, thermal analysis, radiation analysis, and debris analysis. Politecnico di Milano is responsible for mission analysis and GNC. Altec supports the ground segment architecture and interface definition. The Centro Italiano per la Ricerca Aerospaziale (CIRA) is responsible for the execution of the vehicle environmental test campaign. HULD contributes to the development of mission-specific software. VTT is the main payload (ASPECT) provider, and it is supported by the following entities dealing with ASPECT-related development: the University of Helsinki (for the calibration), Reaktor Space Lab (for the development of the Data Processing Unit), Institute of Geology – the Czech Republic Academy of Science (scientific algorithms requirements and testing), and the Brno University of Technology (scientific algorithms development). Finally, INAF-IAPS is the

[36]https://www.mathworks.com/products/simulink.html, last accessed: 18th November 2021.

secondary Payload (VISTA) provider.

The Milani mission has been characterized by a fast development cycle throughout its design. Phase 0 took place during proposal preparation in the Spring of 2020. The Milani team successfully passed the Preliminary Design Review (PDR) in Summer 2021 and the Concurrent Design Review (CDR) in Spring 2022, and it is currently in phase D, as of Summer 2023.

6.2 Image Processing of Milani

Milani's onboard navigation strategy relies on optical observables of D1 extracted from images and then used in an onboard Extended Kalman Filter (EKF). A robust, simple, and accurate IP method is needed. For the case of Milani, information must be extracted from D1 for navigation, but at the same time, D2 must be clearly distinguished in the image for pointing purposes. This derives from D1 being the optimal target for navigation purposes since it is the largest, most visible, and regular body of the binary system. At the same time, D2 is the mission's scientific focus, and therefore it is essential to distinguish it from D1 for dedicated acquisitions.

Moreover, the IP uses data-driven functions, pivoting on the fact that new data about the system could be provided before by the DART and Hera missions. The AOCS subsystem suite is currently designed to host a NavCam (21×16 deg FOV, with a 2048×1536 pixels wide sensor mounted co-axially to ASPECT), a lidar, Sun sensors, a star-tracker, and an inertial measurement unit. The IP will run in the embedded DM3730 processor of the NavCam.

The IP software comprises 5 blocks, as illustrated in Figure 6.5. The input to the IP is a set of data (images, readings from other sensors, variables from higher systems) and configuration parameters (tuning coefficients of the IP functions). The output of the IP is a state vector containing optical observables and quantities extracted from images whose elements are summarized in Table 6.1.

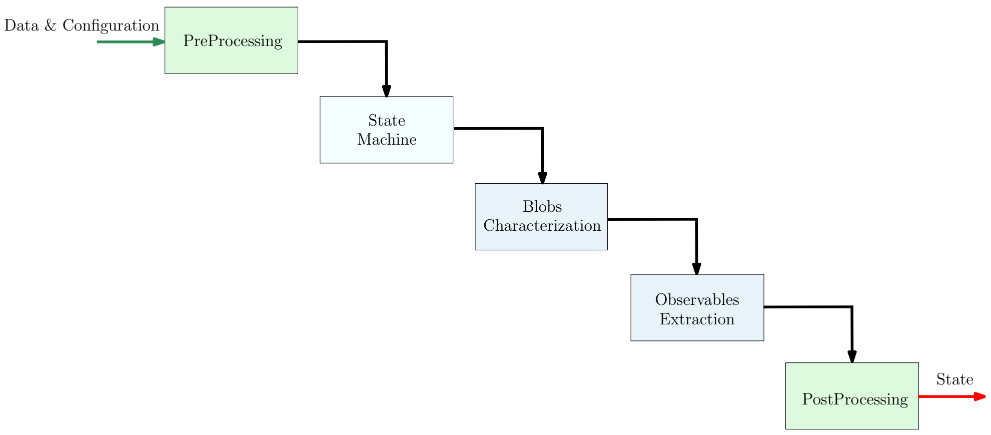


Figure 6.5: High-level architecture of the IP.

Table 6.1: Output vector of the IP.

Name	Symbol	Description
number of bodies	f	Number of bodies detected in the image
CoF of D1	$(CoF)_{D_1}$	Estimated Center of Mass of D1 [37]
CoF of D2	$(CoF)_{D_2}$	Estimated Center of Mass of D2
phase angle	Ψ	Estimated phase angle from D1
range	ρ	Estimated range from D1
ip mode	γ_{ip}	Operative mode of the IP
consistency flag	ν_1	Consistency flag on the output of the IP
asteroid detection	ν_2	Detection flag of a body in the image

The *PreProcessing* and *PostProcessing* blocks handle the interface between the IP and the rest of the onboard software, performing internal logic checks and generating validity or other types of flags. The *State Machine* is the decision-making core of the IP. Based on the validity and operative flags communicated from outside or generated internally, it decides which of the four operative modes γ_{ip} summarized in Table 6.2 to use.

Table 6.2: Operative modes γ_{ip} of the IP, from the lowest to the highest.

Mode	Description
NOP	No operations are performed.
COB	The CoF is estimated as the centroid of the blobs of pixels associated to D1.
WCOB	A data-driven scattering law based only on optical observables is used to generate the *CoF* by correcting the CoB.
SSWCOB	A data-driven scattering law based on optical observables and data from the SS is used to generate the *CoF* by correcting the CoB.

The last three are associated with the choice of the main algorithm to use in the *Observables Extraction* block, while the first is a mode in which no operations are performed. The algorithmic core of the IP resides in the *Blobs Characterization* and *Observables Extraction* blocks, which sequentially process an image and generate optical observables to be used by the rest of the GNC. The task of the *Blobs Characterization* is to generate low-level optical observables from a simple blob analysis while also distinguishing between D1 and D2. The task of the *Observables Extraction* is to further process the image content around D1 to generate a more sophisticated set of observables.

[37]Assuming a homogeneous body, a correct estimate of the CoF would coincide with the CoM. A shift exits if considering a non-homogeneous mass distribution, irrespective of the accuracy of the estimate of the CoF.

6.2.1 Blobs Characterization

Once an image is received by the IP, the first meaningful block in which algorithmic operations are performed is the *Blobs Characterization* one. This block serves a twofold purpose: to distinguish between D1 and D2, and to generate low-level optical observables. The *Blobs Characterization* flowchart is illustrated in Figure 6.6.

The starting point is the image generated by the navcam, which, in the simulation environment, is generated in Blender[38] using CORTO, according to the expected noise characteristics.

At first, the image is binarized using the Otsu method [106] (default choice) or via an arbitrary binary threshold. Morphological operations are then applied to the binary image via a user-defined structuring element in the form of a kernel. Opening, closing, or no operation can be performed on the images, providing great flexibility during operations since the kernel can be easily updated as a configuration parameter of the IP. This step reduces the number of detected blobs in the image, which is helpful in the following blob analysis. The analysis is performed only on the group of pixels larger than a predefined threshold (this is done to remove minor image artifacts) and generates several geometric properties of interest for each blob: e.g., area, bounding box (Γ), centroid coordinates (CoB), eccentricity (e) and major axis length (δ) of the ellipse fitted to the blob of pixels with the same second-order moment. These elements are used to generate a feature vector associated with each blob of pixels.

The feature vectors extracted from the blob analysis are then used to perform object recognition in the image and distinguish between D1 and D2, as well as detect the total number of bodies. The object recognition algorithm is designed as follows:

1. The blobs of pixels are ordered in ascending order based on their areas.
2. The blob with the biggest area is labeled as D1. Its key geometric properties are saved. All other blobs are listed as potential candidates of D2. This is possible assuming that in nominal operation conditions, the blobs of pixel of D1 is always expected to be larger than the one of D2.
3. Γ, the bounding box around D1, is expanded by an arbitrary factor in all directions in the image plane. The expanded bounding box Γ^{ex} is created, represented by the blue dashed rectangle in Figure 6.6.
4. The remaining blobs of pixels which are within Γ^{ex} are removed from the list of D2 candidates. These could be false positive identifications of D2 given by local areas in the terminator region of D1.
5. The biggest blob outside Γ^{ex} is therefore labeled as D2. Its key geometric properties are saved.
6. The number of asteroids detected in the image, f, and the centroid of D2, $(CoF)_{D_2}$, are passed as output of the IP while the other geometrical properties about D1 are passed to the *Observables Extraction* block.

[38] https://www.blender.org/, last time accessed 15th of July 2022.

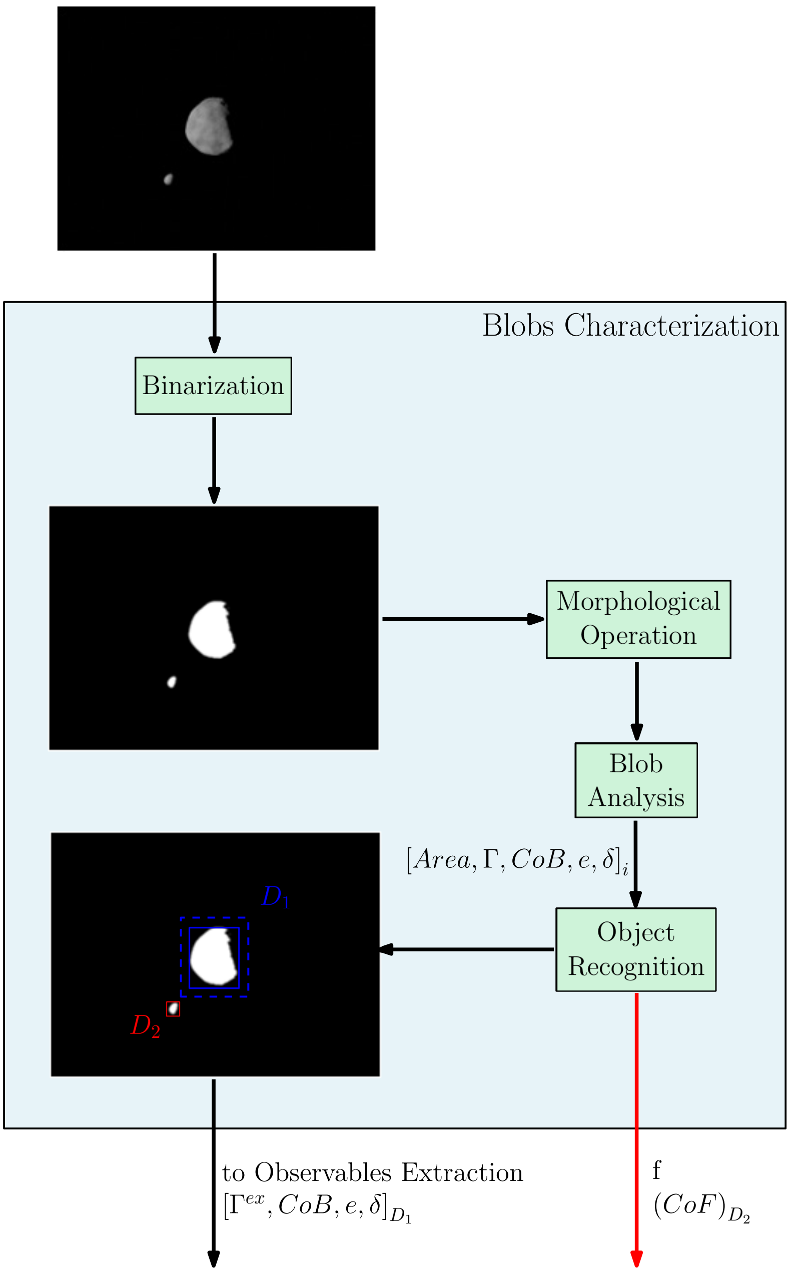


Figure 6.6: High-level architecture of the *Blobs Characterization* block of the IP. The red arrows represent the output of the block that constitutes the components of the state vector.

Note that the CoF of D2 is designed to be equivalent to its CoB. This could not be the case (depending on the operative mode γ_{ip}) for the CoF of D1, as it is illustrated in the next section.

Due to the geometrical configuration of D1, D2, and Milani, several phenomena can occur that impact the visibility of D2. In particular, D2 could be occluded by D1, in the shadow region created by D1, outside of the FOV of the navcam, or in front of D1, occluding portions of it. The latter is the most challenging scenario for the current object recognition algorithm. Some examples of D2 recognition in these scenarios are illustrated in Figure 6.7, together with their associated Γ and Γ^{ex} to exemplify the performance of the algorithm.

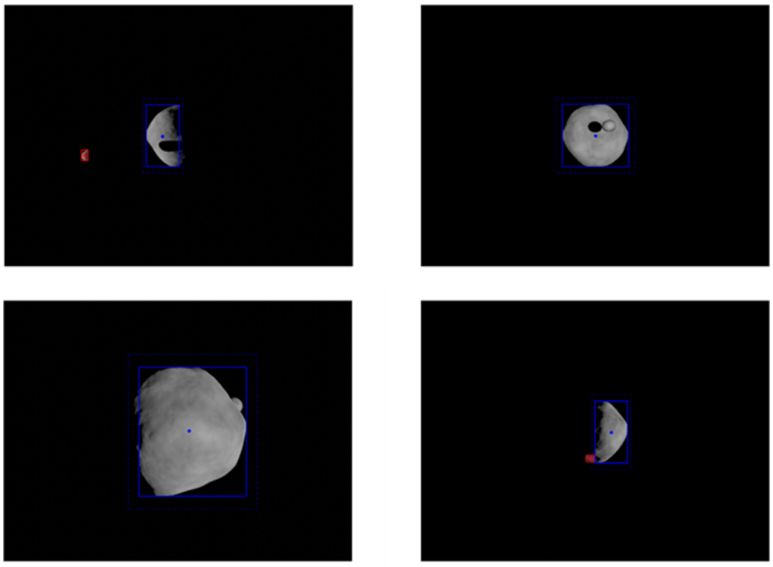


Figure 6.7: Object recognition examples in the Didymos system. Top-right is an example of correct detection of D2, while all other cases are examples of wrong identifications. The top-right and bottom-left cases are missed detections (false negatives). The bottom-right is a false positive identification since a small region of the terminator of D1 gets wrongly identified as D2. The blue and red bounding boxes are associated with D1 and D2, respectively.

6.2.2 Observables Extraction

This block takes as input the Region Of Interest (ROI) of the image around D1 identified by Γ^{ex}, its geometrical properties computed in the *Blobs Characterization* block, and external data to compute high-level optical observables with algorithms that are more sophisticated than a simple blob analysis. The architecture of the *Observables Extraction* block is schematized in Figure 6.8.

Independently from γ_{ip}, the range from D1 is estimated by using a simple apparent diameter relationship:

$$\rho = \frac{R_{D_1}}{\tan\left(\frac{\delta \cdot \zeta}{2}\right)} \tag{6.1}$$

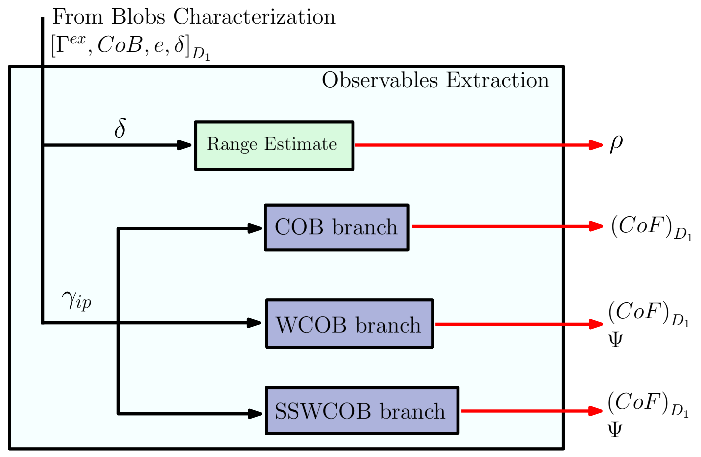


Figure 6.8: High-level architecture of the *Observables Extraction* block of the IP.

where R_{D_1} is the radius of D1 in meters, δ is the major axis length of the blob of pixels, and ζ is the sensor's instantaneous FOV.

The remaining block output is computed based on the operative mode γ_{ip}. These are entangled with the three main algorithms that can be used to calculate the CoF of D1 and the phase angle Ψ: the Center Of Brigthness (COB), Weighted Center Of Brigthness (WCOB), and Sun-Sensor Weighted Center Of Brigthness (SSWCOB). Each constitutes a branch within the *Observables Extraction* block, as illustrated in Figure 6.8.

6.2.2.1 COB

The COB algorithm is a simple, traditional, robust, and well-known method used to estimate the centroid of an object by its center of brightness. The CoB is computed over the binary image in the blob analysis performed in the *Blobs Characterization* block using the following equation:

$$CoB_u = \frac{\sum_{i,j=1}^{N} I_{ij} u_{ij}}{\sum_{i,j=1}^{N} I_{ij}} \quad CoB_v = \frac{\sum_{i,j=1}^{N} I_{ij} v_{ij}}{\sum_{i,j=1}^{N} I_{ij}} \tag{6.2}$$

where I_{ij} is the logic value that determines if the pixel (u_{ij}, v_{ij}) is illuminated or not, whereas CoB_u and CoB_v are the components in pixel of the CoB. The centroid is effectively calculated in the previous block of the *Blobs Characterization* and then passed forward. Note that in this chapter, the notation CoB refers to the centroid of the blobs of pixels, while COB refers to the algorithm branch illustrated in Figure 6.8.

The COB branch does not generate an estimate of Ψ, but generates only an estimate of the CoF of D1. Doing that with the simple centroid formula in Eq. (6.2), introduce a bias given by the irregular shape of the asteroid and the phase angle. To overcome the latter, analytical scattering laws can be used [31, 33]. In the case of the IP of Milani, to overcome this limitation and provide an accurate estimate of the CoM under different geometric conditions while taking advantage of the observation of the system well after the end of the design phase of the algorithm, data-driven scattering laws are applied as variants of the COB.

6.2.2.2 WCOB

The WCOB corrects the CoB by a scattering law derived empirically through data from the target irregular body. The main goal of the WCOB is to generate a correction vector on the image plane that pushes the CoB towards the CoM, assuming a body with constant density. The magnitude and orientation of the correction vector are based solely on geometric observables extracted from images that are used to fit three different data-driven functions. All observables used by the WCOB are rotation and translation invariant.

To design, train, and validate the data-driven functions, a global dataset is generated, referred to as DB_0. The properties of this dataset have varied during different phases of the mission design, as illustrated by the different versions in section A.3.

The purpose of DB_0 is to collect a significant statistical sample of synthetic images generated with CORTO of the Didymos binary system seen from different geometric and illumination conditions. These are tuned to reflect the expected conditions that will be encountered during the FRP and CRP but are also generic enough to be representative of a mission designed to actively observe a small body from the illuminated side.

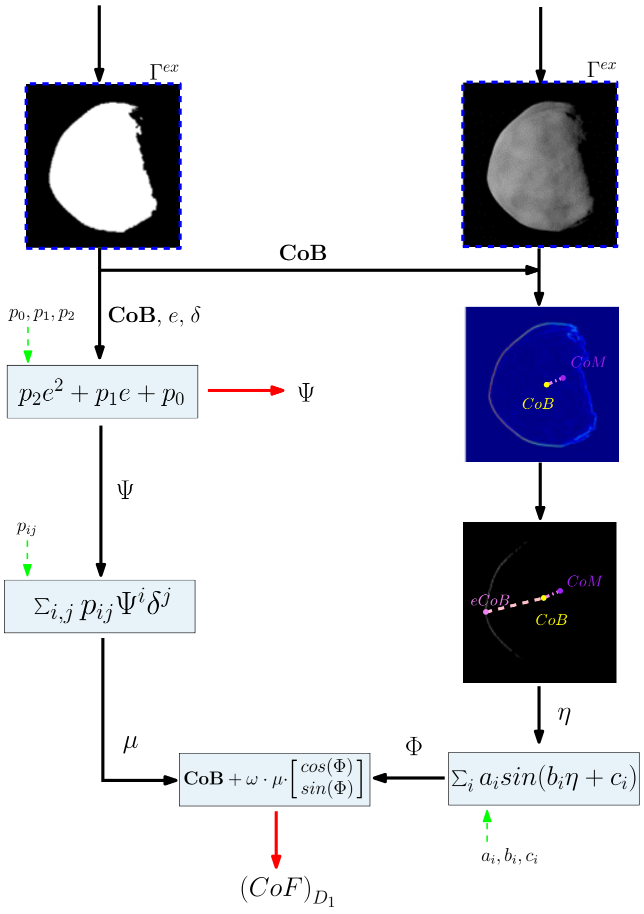
The WCOB is constituted by two pipelines, as it is possible to see in Figure 6.9. These are responsible for the computation of the magnitude and orientation components of the correction term to apply to the CoB of D1. The starting points are two ROIs around D1 taken from the grayscale and binary version of the image corresponding to Γ^{ex}.

Flowing first on the pipeline computing μ (the left one in Figure 6.9), it is possible to determine using the samples in DB_0 that a relationship exists between the eccentricity of the blob of pixels associated to D1 and Ψ. To describe this relationship, a second-order polynomial is used to fit the data represented in Figure 6.10 in the least square sense:

$$\Psi(e) = p_2 e^2 + p_1 e + p_0 \tag{6.3}$$

where p_0, p_1, and p_2 are coefficients evaluated from the fit, while e is the eccentricity of the blob of pixels associated with D1. As illustrated in Figure 6.10, the Ψ estimated with Equation 6.3 would be capable of providing a rough estimate which is more precise at higher values of the phase angle.

Following a similar approach, it is also observed that a relationship could be defined between Ψ, δ, and the difference between the CoF and CoB, the latter being the magnitude of the correction term μ. The relationship is described well by



Γ^{ex}
Γ^{ex}
CoB
p_0, p_1, p_2
CoB, e, δ
$p_2e^2 + p_1e + p_0$
Ψ
CoM
CoB
Ψ
p_{ij}
$\Sigma_{i,j}\, p_{ij}\Psi^i\delta^j$
eCoB
CoM
CoB
η
μ
Φ
$\mathbf{CoB} + \omega \cdot \mu \cdot \begin{bmatrix} cos(\Phi) \\ sin(\Phi) \end{bmatrix}$
$\Sigma_i\, a_i sin(b_i\eta + c_i)$
a_i, b_i, c_i
$(CoF)_{D_1}$

Figure 6.9: Schematic of the WCOB algorithm. The red arrow represents the output. The green ones are the configuration parameters needed by the data-driven functions.

a fifth-order polynomial surface, which is once again fit in the least squares sense using the samples of DB_0:

$$\mu(\Psi,\delta) = \sum_{\substack{i=0,\cdots,5 \\ j=0,\cdots,5 \\ i\cdot j\leq 6}} p_{ij}\,\Psi^i\delta^j \tag{6.4}$$

Switching to the second pipeline of the WCOB algorithm, Φ is computed starting from the ROI of the grayscale image around D1. A filter is applied to exacerbate differences between the soft and sharp gradient over the terminator and edge of the asteroid. To do so, a Sobel filter is used. Once the activation map of the Sobel filter is generated, an arbitrary factor is used to threshold the map by a fraction of the maximum value present in it. The generated binary map is then analyzed, and the CoB of the largest blob of pixels associated with the region on the edge of D1, is computed. This is referred to as $eCoB$ (edge CoB) and is used to provide information about the lighting conditions on the asteroid. The

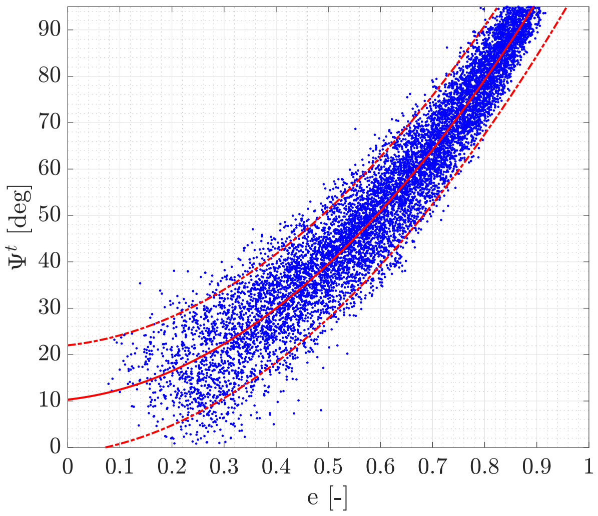


Figure 6.10: *Ψ* function (solid red) and *3σ* value (dashed red) together with all datapoints of *DB₀* (blue).

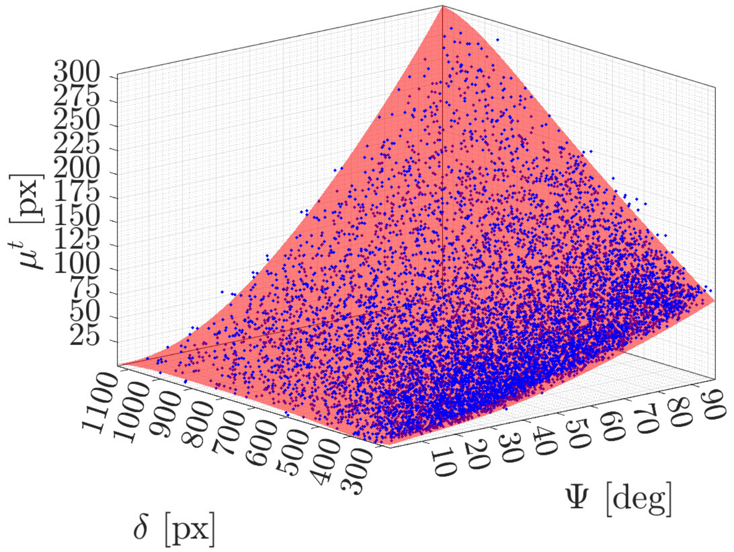


Figure 6.11: μ function (red surface) vs *Ψ* and δ for all datapoints of DB_0 (blue).

CoM, *CoB* and *eCoB* are illustrated in one case in Figure 6.9. From this figure, it is possible to see that the line connecting the *eCoB* with the *CoB* can be used to estimate the orientation of the line connecting the *CoB* with the *CoM*, which represents the ultimate orientation at which the correction term of the WCOB method should aim. The *eCoB* is then used with the *CoB* of D1 to compute an orientation in the image plane, referred to as η. Once again, by plotting the estimated orientation η with the true one *Φ* for the samples in DB_0, a relationship can be seen between these quantities, as illustrated in Figure 6.12. The following equation represents this relationship:

$$\Phi(\eta) = \sum_{i=1,\cdots,4} a_i sin(b_i\eta + c_i) \tag{6.5}$$

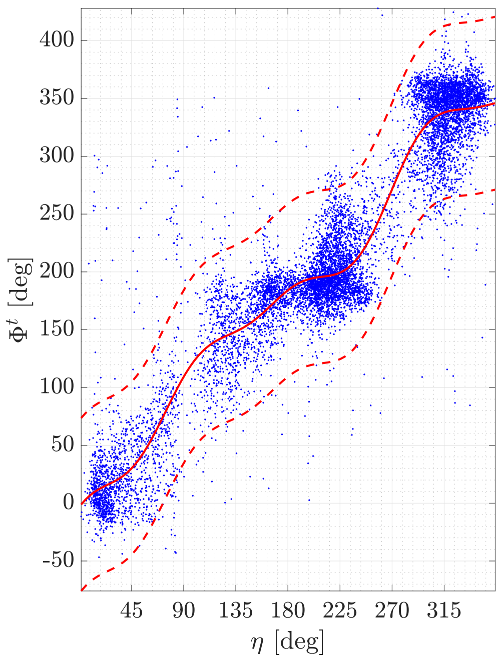


Figure 6.12: Φ function (solid red) and 3σ value (dashed red) together with all datapoints of DB_0 (blue).

Now that both μ and Φ have been computed from the image, they are combined in the following equation:

$$\begin{bmatrix} CoF_u \\ CoF_v \end{bmatrix} = \begin{bmatrix} CoB_u \\ CoB_v \end{bmatrix} + \omega \cdot \mu(\Psi(D), \delta) \cdot \begin{bmatrix} \cos(\Phi(D)) \\ \sin(\Phi(D)) \end{bmatrix} \tag{6.6}$$

where ω is a weighting factor used to tune the magnitude of the correction term. It is immediate to understand that when $\omega = 0$ the WCOB degenerates into the COB. By default, a value of $\omega = 1$ is used for all geometric conditions, but in general, this parameter could be optimized or varied in real time depending on the operative conditions.

6.2.2.3 SSWCOB

A similar approach is employed by the SSWCOB, with the significant difference that data from the SS is combined with data extracted from the image. The main advantage lies in the high accuracy that can be achieved in predicting Ψ, which is better than the one extracted from the image, positively influencing the performance of the whole branch.

As for the WCOB, the main goal of the SSWCOB is to generate the same correction vector on the image plane that pushes the *CoB* towards the *CoM*. Oppositely from the WCOB, a large portion of the functionalities and quantities extracted from the image are substituted by data from the Sun Sensor. This simplifies the algorithm but makes it dependable on additional sensors. The architecture of the SSWCOB method is illustrated in Figure 6.13.

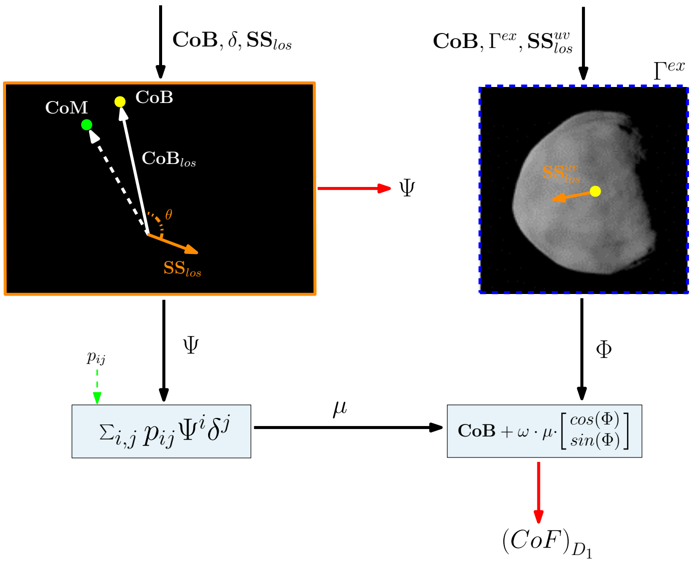

$\mathbf{CoB}, \delta, \mathbf{SS}_{los}$
$\mathbf{CoB}, \Gamma^{ex}, \mathbf{SS}_{los}^{uv}$
Γ^{ex}
CoM
CoB
CoB_{los}
θ
SS_{los}
SS_{los}^{uv}
Ψ
Ψ
Φ
p_{ij}
$\Sigma_{i,j}\, p_{ij} \Psi^i \delta^j$
μ
$\mathbf{CoB} + \omega \cdot \mu \cdot \begin{bmatrix} cos(\Phi) \\ sin(\Phi) \end{bmatrix}$
$(CoF)_{D_1}$

Figure 6.13: Architecture of the SSWCOB algorithm. The red arrow represents the output of the algorithm.

As for the WCOB, two pipelines are identified for the determination of μ and Φ. Flowing first from the μ pipeline, data retrieved from the Sun sensor is used to compute Ψ. The IP receives the line of sight vector of the Sun direction in the CubeSat reference frame estimated by the Sun sensor. Using known rigid rotation matrices and assuming to know the attitude quaternion from the ADCS subsystem, the CoB_{los} and SS_{los} are transformed in the same reference frame. The angle θ between these two lines of sights is therefore computed and related to Ψ as:

$$\Psi \approx \pi - \theta \tag{6.7}$$

This formula is an approximation of Ψ since the computation is performed from the *CoB* (available from the image) and not from the *CoM* (unknown at the moment of the estimation). As for the WCOB, once Ψ is determined, μ is computed by applying Equation 6.4.

Switching now to the pipeline to compute Φ, the projection of the line of sight of the Sun in the image plane is used, SS_{los}^{uv}. This quantity, centered on the *CoB* of D1, is used to provide an orientation from which the angle Φ is computed.

Having determined both μ and Φ, the same formula used in the WCOB is now applied to determine the correction term to apply to the CoB of D1:

$$\begin{bmatrix} CoF_u \\ CoF_v \end{bmatrix} = \begin{bmatrix} CoB_u \\ CoB_v \end{bmatrix} + \omega \cdot \mu(\Psi, \delta) \cdot \begin{bmatrix} \cos(\Phi(D)) \\ \sin(\Phi(D)) \end{bmatrix} \tag{6.8}$$

with respect to Equation 6.6, now only the orientation function Φ is dependant on dataset quantities from DB_0.

6.3 Guidance, Navigation, and Control subsystem of Milani

The GNC of Milani can be described as a semi-autonomous, vision-based subsystem with the main task to provide a primary pointing to the ADCS. During most of the mission, this coincides with the pointing of the payload deck of the CubeSat, corresponding to the face on which navcam, LiDAR, and ASPECT are co-axially mounted. However, in selected circumstances, it may be desired to direct one of the ISL antennas towards Hera or to achieve a different pointing.

Furthermore, the GNC estimates the spacecraft state, intended as its position and velocity in the inertially fixed *ECLIPJ2000* reference frame, centered on the system barycenter. In addition to being used for navigation purposes, this estimate can generate the primary pointing from simple geometric considerations.

The GNC capabilities are enabled by the advanced functioning of the IP. However, by design, the GNC subsystem does not have authority over the translational guidance and control computed from the ground and uplinked to the CubeSat. A dedicated thrusters management module handles the execution of the loaded maneuvers.

The architecture of the GNC is composed of 5 blocks, as illustrated in Figure 6.14, whose tasks are similar (by shared design choices) to those already presented for the IP in Section 6.2. The first is the *PreProcessing* block, which performs initial checks on the input variables to ensure their validity. These include freshness checks to ensure that incoming signals have been updated recently and integrity checks to verify that the values are received within expected intervals. After that, the *State Machine* determines the appropriate operative mode based on predefined logic. Then, the *Navigation* and *Guidance* blocks follow, in which optical observables and onboard ephemerides are used to compute the desired pointing profile. The validity of the navigation and guidance output is verified in the *Health check* block before being provided to the rest of the system as an output state vector.

The functionalities of the GNC are defined by five different operative modes γ_{gnc}, briefly described in Table 6.3. These are also devised to communicate the status of the GNC to other systems.

The modes are selected using a series of truth tables that inspect logic conditions based on the input signals and the checks computed within the *PreProcessing*

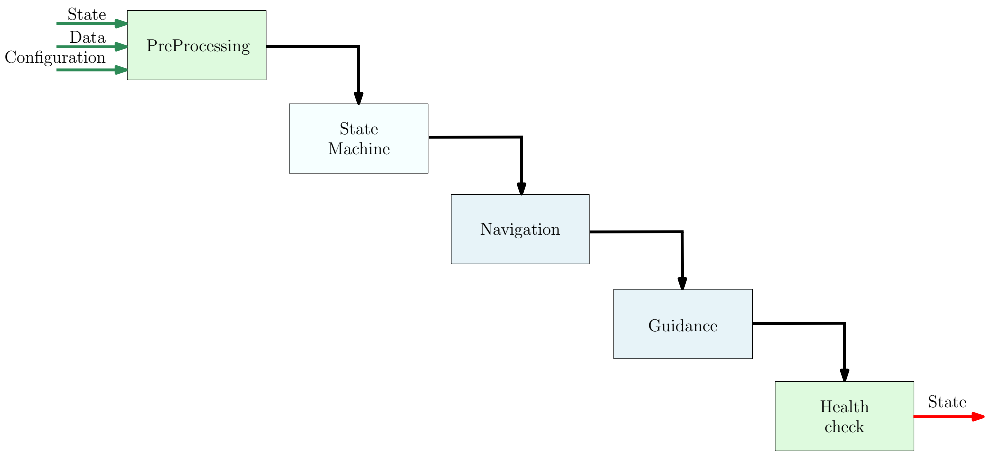


Figure 6.14: High-level architecture of the GNC.

Table 6.3: Operative modes γ_{gnc} of the GNC.

Mode	Description
Drift	This mode is triggered when an issue has been detected in *Navigation* or *Guidance* blocks. In such a case, neither output of these blocks is considered reliable.
Navigation	The best navigation strategy is targeted for execution. The output of the *Guidance* block is unreliable.
Guidance	The best guidance strategy is targeted for execution. The output of the *Navigation* block is unreliable.
Nominal	Both *Navigation* and *Guidance* blocks are executed targeting the best possible strategy. Their outputs are considered reliable.
Asteroid Search	Ephemeris-based navigation is targeted whenever possible. The guidance of the primary pointing is designed to re-acquire the target in the FOV.

block. In all cases, the default mode is always the simplest; if the necessary conditions are met, the system automatically advances to more complex ones. The only exception is represented by the *Asteroid Search* mode, which is intended as a contingency option and, therefore, requires ground intervention and is not autonomously activated onboard. It is also noted that the highest reachable mode can be limited from the ground using a set of configuration parameters.

A generic overview of the GNC design before CDR is briefly illustrated in [157], while a detailed version is described in [158]. The following sections briefly describe the functioning of the *Navigation* and *Guidance* blocks.

6.3.1 Navigation

In the *Navigation* block, the state of the CubeSat with respect to the asteroid system is estimated. The algorithms used in this block are driven by three different navigation sub modes γ_{nav}: *Navigation keep last*, *Navigation from ephemerides*, and *Navigation from EKF*.

In *Navigation keep last*, the navigation solution is not updated, and the previous solution produced is kept as output. In *Navigation from ephemerides*, ephemeris data provided from the ground and stored as Chebyshev polynomial coefficients are interpolated to estimate a navigation solution. Lastly, in *Navigation from EKF*, the navigation solution is provided by the onboard EKF, which relies on optical observables from the IP as well as ranging data from the LiDAR to generate a state estimate.

The onboard implementation of the EKF uses a dynamical model accounting for the point-mass gravitational effects of D1, D2, and the Sun, as well as for the SRP, which is modeled with a simple cannonball model. The SRP acceleration is split into a deterministic and a stochastic component. A stochastic residual acceleration is also included [158] to account for other uncertainties in the dynamic model.

The equations of motion of the filter are propagated using a Runge-Kutta 4^{th} order integrator, while the State Transition Matrix (STM) is computed onboard using a second-order approximation. The filter is designed to take as input the CoF of D1 and the range measurement from the LiDAR. Other measurements are discarded at the current design stage since they are not considered sufficiently reliable. The measurement update of the CoF is set to 30 minutes while the range from the LiDAR is set to five minutes, if available. Both the IP and LiDAR work within a specific range envelope from D1: the former is designed to work between 3 and 23 km, while the latter is assumed to work below 5.5 km. Measurements are considered to be affected by Gaussian random noise with 0 mean and a standard deviation of 15 m for the LiDAR (accounting both the error due to the sensor and the uncertainty of the shape), and 40 px, 20 px, and 15 px respectively for the COB, WCOB, and SSWCOB. More details about the filter are described in [158].

6.3.2 Attitude Guidance

By choice, Milani does not possess onboard translational guidance and control capabilities. These otherwise traditional capabilities of a GNC subsystem are demanded and computed on the ground and then uplinked to the CubeSat. On the other hand, the attitude guidance profile of the primary pointing is generated onboard. This task is performed within the *Guidance* block and can be executed by five different strategies: *Guidance keep last*, *Reference*, *Tracking*, *Predicted*, and *Search pattern*.

In *Guidance keep last*, the last computed guidance solution is used. In *Reference*, the guidance is obtained from ground-based information, either by following a specified pointing profile or interpolating ephemerides data. In *Tracking*, data from the IP is used to track the target asteroid and keep it at the center of the navcam FOV. A fixed inertial pointing is kept while waiting for new IP data. In *Predicted*, the position estimated by the EKF is combined with the ephemerides of the target body to compute a pointing solution. While in *Tracking*, the target must be detectable by the IP, which limits it to either D1 or D2, in *Predicted*, it can be any geometric point in the system. Finally, in *Search pattern*, a contingency guidance strategy is implemented to recover the target body in the navcam FOV after it has been lost. In this strategy, the primary pointing is computed by following a profile obtained from a predefined map while traversing the space, avoiding a Sun-exclusion cone. When the target is positively detected, the guidance strategy automatically switches to *Tracking*.

6.4 Performance

A set of extensive analyses has been performed to validate the design of the IP and GNC with a twofold objective: to confirm the expected behavior of the systems during nominal and off-nominal events (in particular attitude and orbital maneuvers, sensor faults, missing input data, and other contingency scenarios), and to assess compliance with performance requirements. This campaign has been carried out throughout the development of the mission from phase 0 in the Summer of 2020 to CDR in the Summer of 2022. This campaign consisted of both open-loop and closed-loop high-fidelity simulations performed in Matlab/Simulink, with the simulation framework developed by the DART group for proximity operation scenarios.

In the remainder of this section, a small subset of illustrative examples of the performance achieved by IP and GNC is illustrated, as listed below:

1. A global static assessment of the performance of the object recognition algorithm.
2. A global static assessment of the IP is presented for both the FRP and CRP phases of the mission.
3. An example of a typical pointing performance assessment is illustrated on arc 4b of the FRP.
4. An example of a typical position estimation performance assessment is illustrated on arc 4b of the FRP.

Arc 4b of the FRP is chosen since it represents a challenging traverse over the Didymos environment in which the CubeSat gets very close to the system.

6.4.1 Performance of the IP

In this section, the performance of the IP is assessed using the $\mathcal{DS}_7$ and $\mathcal{DS}_8$ datasets described in section A.3.

6.4.1.1 Object recognition

The performance of the object recognition algorithm illustrated in Section 6.2.1 are assessed for the dataset $\mathcal{DS}_7$ and its subsets ($\mathcal{DS}_7^{DB_0}$, $\mathcal{DS}_7^{FRP}$, and $\mathcal{DS}_7^{CRP}$) using the metrics defined in Section 1.8.2.

The values of accuracy, precision, and recall are reported in Table 6.4 for the three datasets[39]. First, it is possible to see that the conditions in which D2 is observed are well balanced among the different splits of $\mathcal{DS}_7$. By dataset design, D1 is always visible in the samples of these datasets. Thus, the metrics are essential in evaluating the correct detection of D2.

Table 6.4: Performance of the object recognition algorithm.

Metric	$\mathcal{DS}_7^{DB_0}$	$\mathcal{DS}_7^{FRP}$	$\mathcal{DS}_7^{CRP}$
A	91.20	87.39	86.15
P	99.85	99.95	99.83
R	89.27	85.86	81.77

From a performance perspective, the object recognition is capable of recognizing D2 with high precision (≥ 99.83 in all datasets) but with a medium-high recall ($81.77 \leq R \leq 89.27$). Therefore, the object recognition algorithm is characterized by a low rate of FP and a high rate of FN detection of D2.

Such performance is a consequence of the algorithm design, which for robustness is based on the concept of an expanded bounding box rejecting regions of the terminator of D1 from being wrongly labeled as D2. However, this step also introduces a high rate of FN whenever D2 is transiting above D1. This behavior has been noted and accepted once it has been observed to impact minimally over the performances of the other IP algorithms, and with the philosophy not to complicate the design of the algorithm.

The only rare scenario in which the IP algorithms have been observed to be significantly impacted by the disturbance generated by D2 is when the two bodies are as far away as possible from each other, and their edges are connecting, generating an integrated blob of pixels much greater than expected.

6.4.1.2 Centroid, phase angle, and range regression

This section discusses the performance of the *Observables Extraction* block of the IP. The assessment is focused on the $(CoF)_{D_1}$, Ψ, and ρ output of the IP thus excluding the $(CoF)_{D_2}$ from the analysis. Performance is assessed both on $\mathcal{DS}_7$ and $\mathcal{DS}_8$, in this order, considering the metrics defined in Section 1.8.1.

The SSWCOB is evaluated without introducing any error in the attitude knowledge nor in the line of sight reading from the Sun Sensor, which is therefore

[39]note that the object recognition algorithm does not require a training set, thus DB_0 is used as a test set in this case

modeled as an ideal sensor. This choice has been made to have the SSWCOB being purely evaluated from an IP perspective and at the best of its capability. For a more realistic evaluation, its performances shall be modeled with multiple sets of errors both in the attitude and sensor readings.

First, a series of histograms illustrate the distributions of ε^{n}_{CoF}, ε_{ψ}, and ε_{ρ} in the $\mathcal{DS}^{FRP}_{7}$ and $\mathcal{DS}^{CRP}_{7}$ datasets [40].

From the histograms in Figure 6.15, it is possible to appreciate the beneficial effect of the scattering laws of the WCOB and SSWCOB in increasing the accuracy of the CoF estimate. In Figure 6.16, it is possible to see that Equation 6.7, although an approximation, is capable of estimating Ψ much more accurately than the estimate generated from the image alone. This result has been expected and is reflected in the performance of the WCOB and SSWCOB methods: with an accurate Ψ, the resulting μ is estimated better since both the WCOB and SSWCOB are evaluated over the same δ for each image.

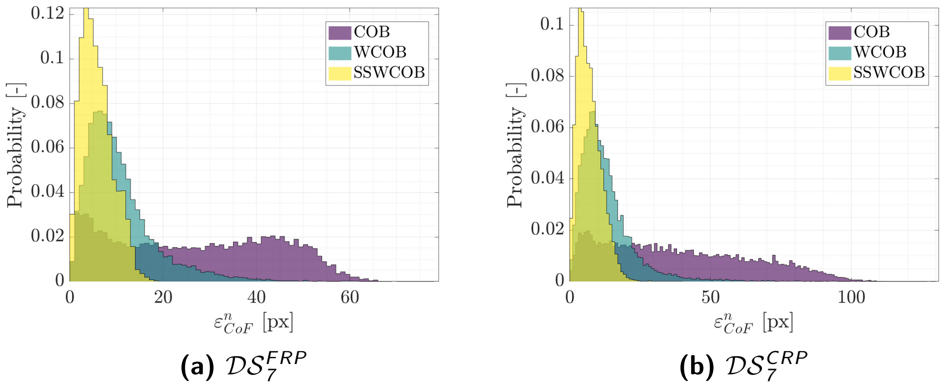
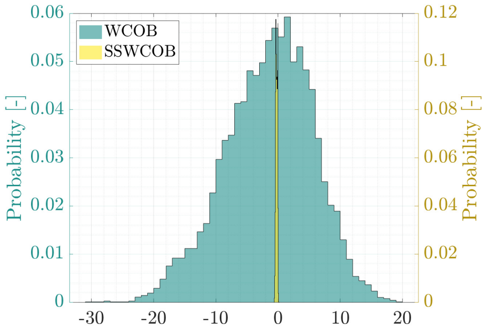


Figure 6.15: Histograms of the ε^{n}_{CoF} errors of the COB, WCOB, and SSWCOB. The width of the bins is set to *1* pixel.

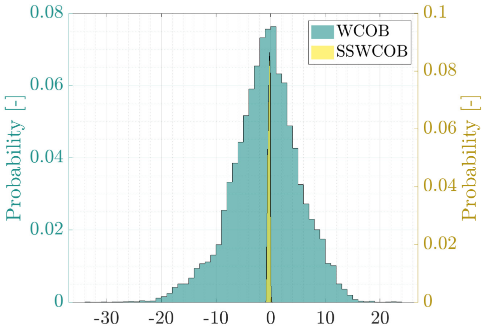
Looking at the histograms of ε_{ρ} in Figure 6.17, it is possible to conclude that the range is estimated with a considerable bias and a high standard deviation. This makes the predicted range unreliable for standalone use, e.g., triggering an event or another functionality outside the IP. The range error is also represented as function of relative percentage error with respect to the true range in Figure 6.18.

Another interesting visualization of the performance is visible in Figure 6.19, depicting the distributions of ε^{u}_{CoF} and ε^{v}_{CoF} in the image plane, together with their error ellipses. It is possible to visually appreciate what the previous histograms have already illustrated: a trend in the increase of the accuracy passing from the COB to the WCOB to the SSWCOB. It is also possible to note the different orientations of the ellipses from $\mathcal{DS}^{FRP}_{7}$ to $\mathcal{DS}^{CRP}_{7}$ and a bias in the CoF estimate by the COB method in the $\mathcal{DS}^{CRP}_{7}$, which is explained by the nature of its asymmetrical

[40] Note that the same color code is used across this section to distinguish the performance of the COB, WCOB, and SSWCOB

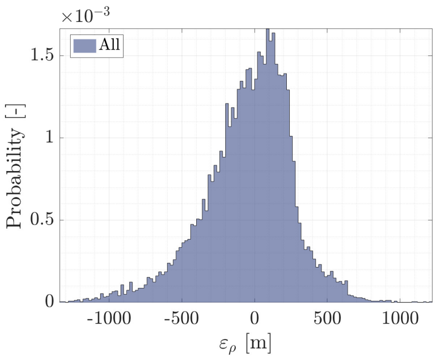


(a) $\mathcal{DS}_7^{FRP}$

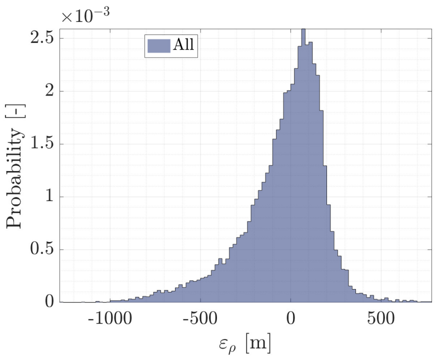


(b) $\mathcal{DS}_7^{CRP}$

Figure 6.16: Histograms of the ε_ψ errors of the WCOB, and SSWCOB. The width of the bins is set to *1*° and *0.05*° respectively, for the WCOB and SSWCOB.

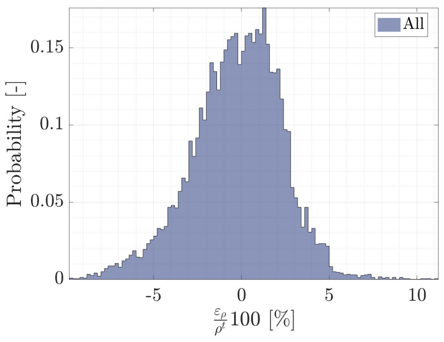


(a) $\mathcal{DS}_7^{FRP}$

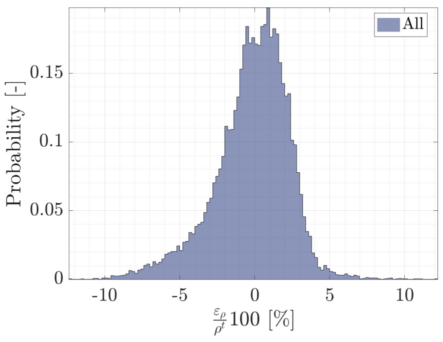


(b) $\mathcal{DS}_7^{CRP}$

Figure 6.17: Histograms of the ε_ρ errors. The width of the bins is set to *20* m.



(a) $\mathcal{DS}_7^{FRP}$



(b) $\mathcal{DS}_7^{CRP}$

Figure 6.18: Histograms of the relative percentage error of ε_ρ. The width of the bins is set to *0.5*%.

Table 6.5: Performance metrics of the COB, WCOB, and SSWCOB in the $\mathcal{DS}_7^{FRP}$ and $\mathcal{DS}_7^{CRP}$ datasets.

		$\mathcal{DS}_7^{FRP}$			$\mathcal{DS}_7^{CRP}$		
	Metric	COB	WCOB	SSWCOB	COB	WCOB	SSWCOB
ε^n_{CoF}	μ [px]	27.00	10.60	5.87	38.46	12.69	6.96
	σ [px]	17.05	7.08	3.62	25.52	8.69	4.41
ε^u_{CoF}	μ [px]	-2.47	1.12	-0.41	-34.70	-0.64	0.66
	σ [px]	29.66	7.28	3.71	28.38	8.69	4.73
ε^v_{CoF}	μ [px]	-2.22	-2.10	-0.75	-6.74	1.95	-0.58
	σ [px]	11.37	10.49	5.58	8.67	13.22	6.52
ε_ψ	μ [deg]	n.a.	-1.41	-0.23	n.a.	-0.95	-0.31
	σ [deg]	n.a.	7.03	0.15	n.a.	6.12	0.21
ε_ρ	μ [m]	-45.19	-45.19	-45.19	-37.08	-37.08	-37.08
	σ [m]	302.64	302.64	302.64	225.30	225.30	225.30

trajectories, as illustrated in Figure 6.3. The parameters of the error ellipses in Figure 6.19 are reported for completeness in Table 6.6.

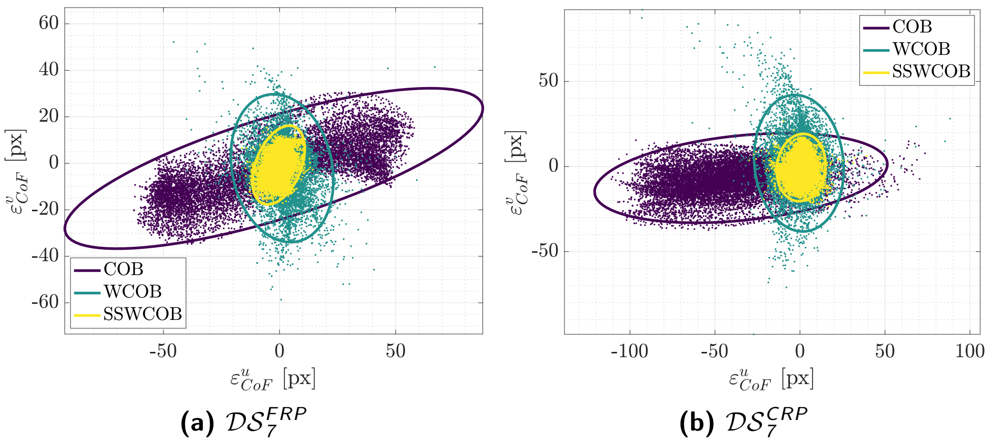


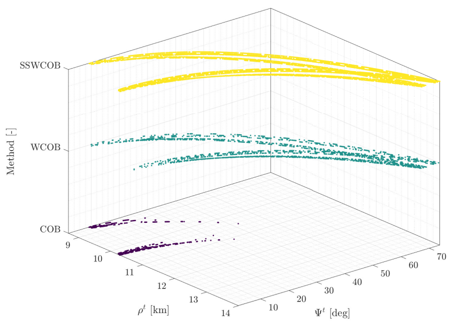
Figure 6.19: Error ellipses in image plane of ε^u_{CoF} and ε^v_{CoF} for the COB, WCOB, and SSWCOB methods. A *99%* confidence interval is used to draw the ellipses.

Figure 6.20(a) and Figure 6.20(b) illustrate the best method for each phase identified as the one achieving the smallest ε^n_{CoF}. The points on the $\mathcal{DS}_7^{FRP}$ and $\mathcal{DS}_7^{CRP}$ datasets associated with the best methods are represented in a phase space with Ψ^t and ρ^t. In the $\mathcal{DS}_7^{FRP}$ the COB, WCOB, and SSWCOB are considered the best respectively for *8.71%*, *22.94%*, and *68.35%* of the cases. In the $\mathcal{DS}_7^{CRP}$ the COB, WCOB, and SSWCOB are considered the best respectively for the *8.18%*, *22.18%*, and *69.64%* of the cases above *4* km. From Figure 6.20(a)

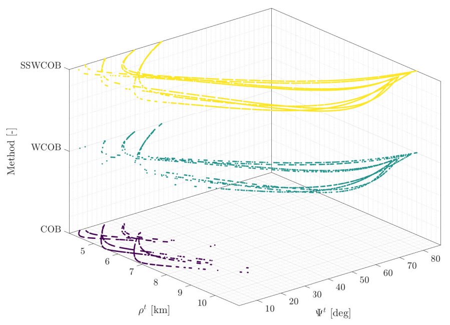
Table 6.6: Parameters of the error ellipses from Figure 6.19.

	$\mathcal{DS}_7^{FRP}$			$\mathcal{DS}_7^{CRP}$		
Parameter	COB	WCOB	SSWCOB	COB	WCOB	SSWCOB
X_0 [px]	-2.47	1.12	-0.41	-34.70	-0.64	0.66
Y_0 [px]	-2.22	-2.10	-0.75	-6.74	1.95	-0.58
a [px]	93.89	32.20	17.85	86.55	40.23	19.88
b [px]	21.86	21.53	9.75	24.92	26.18	14.24
θ_e [deg]	197.00	101.83	67.87	185.85	95.77	82.31

and Figure 6.20(b) is also possible to see a clear preference for the COB method whenever Ψ^t is low. Applying a data-driven scattering law in these cases seems unfruitful compared to a simple CoB estimate.



(a) $\mathcal{DS}_7^{FRP}$



(b) $\mathcal{DS}_7^{CRP}$

Figure 6.20: Scatter plot of the method with the smallest ε_{CoF}^n as a function of the range and phase angle in $\mathcal{DS}_7^{FRP}$ (a) and $\mathcal{DS}_7^{CRP}$ (b).

It is also interesting to visualize in Figure 6.21 the same plot in the phase space only for those points below 4 km, which are outside the training envelop of the WCOB and SSWCOB methods. It is possible to see that in this range, there are points outside this envelope that are still providing better solutions than the COB method, but also that the COB method is the preferred option whenever the body saturates the FOV of the NavCam below 4 km. The latter is an artifact given by the ideal pointing to generate the images since saturated images tend to have a centered CoB.

Figure 6.22 and Figure 6.23 attempt to represent the points in position space for the $\mathcal{DS}_7^{DB_0}$ in which the WCOB is better than the COB and the SSWCOB is better than the WCOB, using as metric the ε_{CoF}^n error. From both figures, it is immediately clear that the WCOB is exceptional in reducing substantial errors in a wide range of phase angles from medium to high, being the illumination from the Sun coming from the X-axis. At the same time, it is possible to see that the gain

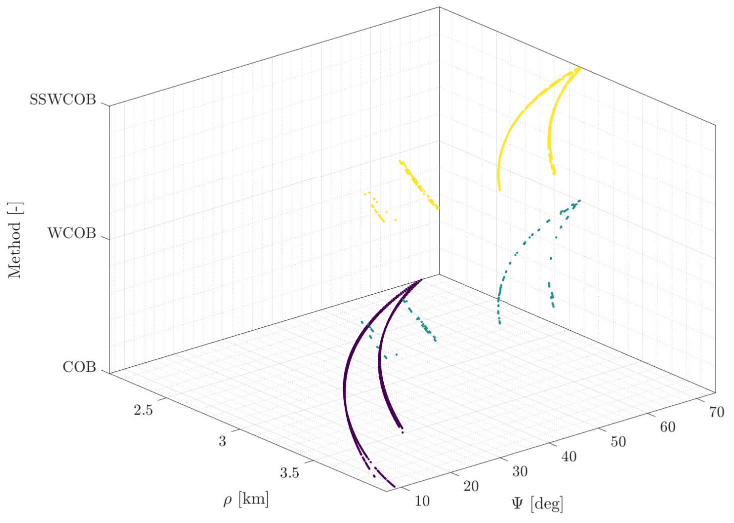
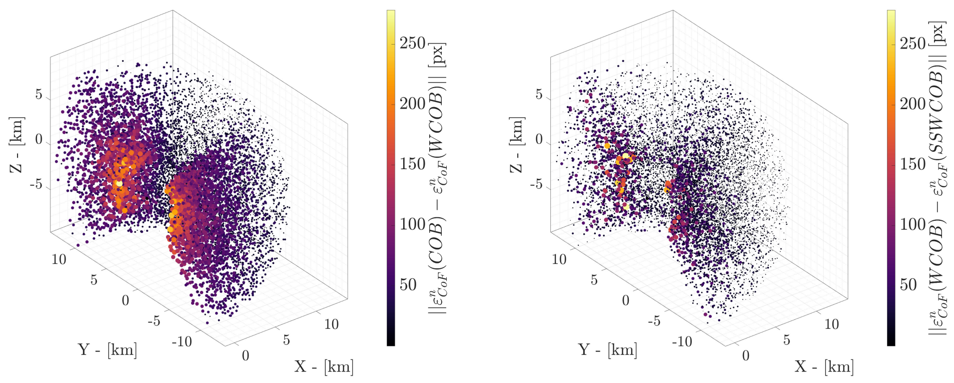


Figure 6.21: Scatter plot of the method with the smallest ε^n_{CoF} as a function of the range and phase angle in $\mathcal{DS}^{CRP}_7$ for the points below 4 km.

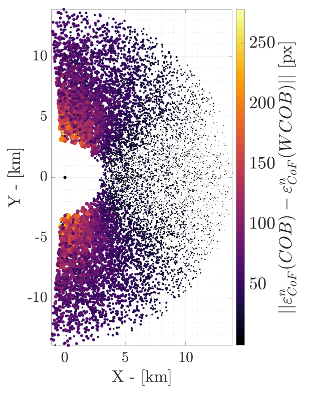
of the SSWCOB over the WCOB is less remarkable if not in the areas at very high phase angles while is being consistently spread across various points in space for smaller improvements.



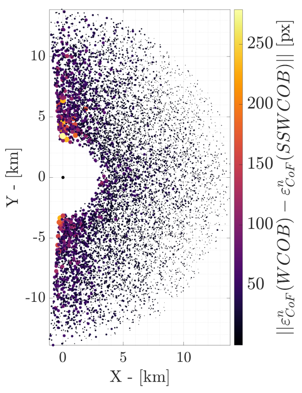
(a) COB-WCOB **(b)** WCOB-SSWCOB

Figure 6.22: 3D view of the scatter plot of the error between the COB and WCOB (a) and WCOB and SSWCOB (b) in the $\mathcal{DS}^{DB_0}_7$ dataset. The color metric used is the difference between the ε^n_{CoF} errors.

In Figure 6.24 and Figure 6.25, it is possible to see the performance of the methods as a function of time during the first two arcs of the FRP and CRP, together with the values of ρ^t and Ψ^t.



(a) COB-WCOB



(b) WCOB-SSWCOB

Figure 6.23: Top view of the scatter plot of the error between the COB and WCOB (a) and WCOB and SSWCOB (b) in the $\mathcal{DS}_7^{DB_0}$ dataset. The color metric used is the difference between the ε^n_{CoF} errors.

The estimate of Ψ by the WCOB seems to degrade with low values of Ψ^t. Indeed, around days *2* and *5.5* of the FRP and days *2.5* and *3.75* of the CRP it is possible to see the COB outperforming all methods in points in which there are low values of Ψ^t. On the other hand, the error on the range varies with ρ^t with an expected trend, being affected more from ρ^t than Ψ^t.

While for most of the phases, low values of Ψ^t are associated with low values of ρ^t and vice-versa, around day 3 of the CRP an interval of time is visible in which this is not true. In this interval, the performance of ρ follows the trend driven by ρ^t. Being the range estimated using δ, as described in Equation 6.1, it is possible to conclude that this parameter is capable of providing a robust estimate in the face of challenging illumination conditions. This is a consequence of a geometric property of the fitted ellipse in the image plane: assuming a constant distance while facing D1 with varying phase angles, the variability of δ would be negligible.

Finally, from Figure 6.25, it is also possible to see a limitation of data-driven approaches of important consequences from an operational point of view: outside their training envelope, the methods cannot be used. Because this scenario may happen, the COB is designed to replace the WCOB and SSWCOB whenever conditions for their applications are not met.

The performance of the IP are also presented for a different iteration of the trajectories in the FRP and CRP, considered in the $\mathcal{DS}_8$ dataset. The performance of the various IP modes in the two testing datasets of $\mathcal{DS}_8^{FRP}$ and $\mathcal{DS}_8^{CRP}$ in terms of ε_α, ε_ψ, and ε_ρ are summarized in Table 6.7.

From the values of ε_α, similar conclusions to those already discussed in the previous part of the section can be drawn about the difference in performance between COB, WCOB, and SSWCOB.

For completeness, the performance of ε_α for the three strategies over the entire

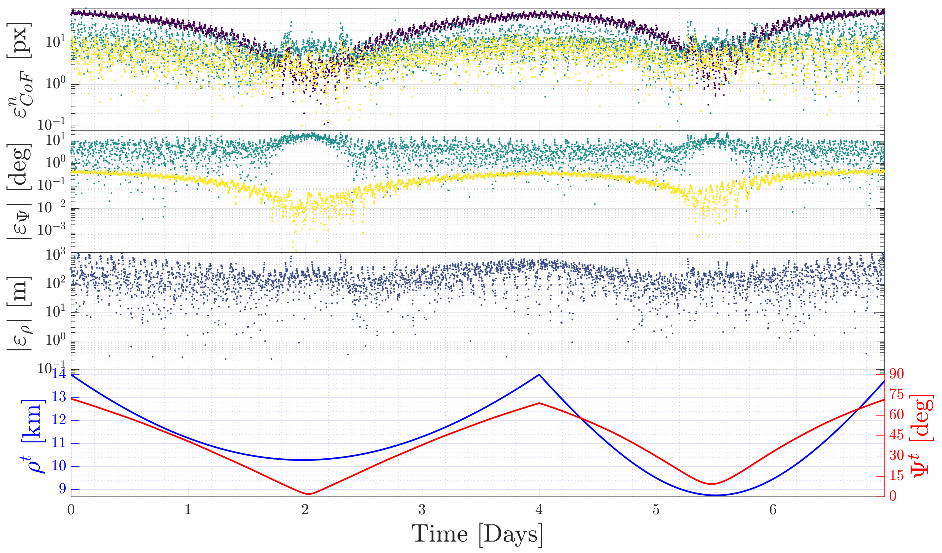


Figure 6.24: Performances of the COB, WCOB, and SSWCOB as function time during the first two arcs of the FRP.

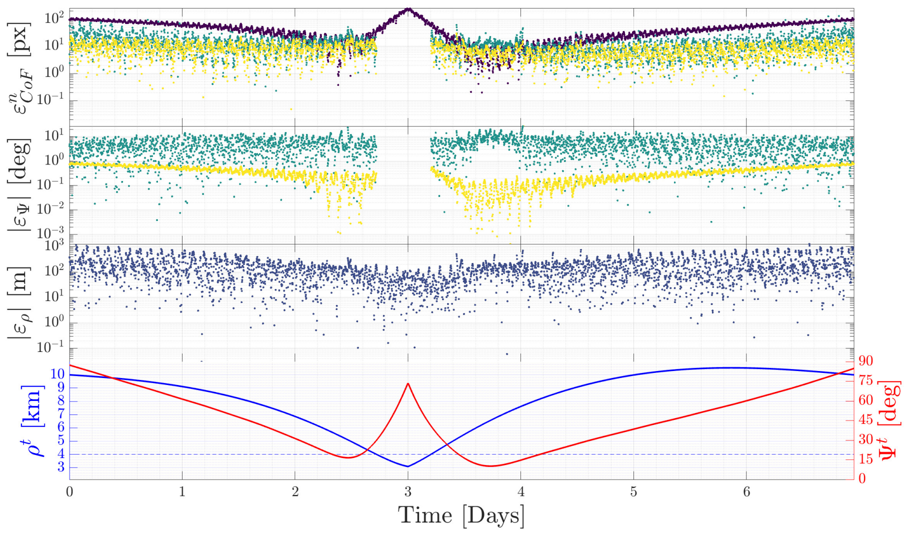


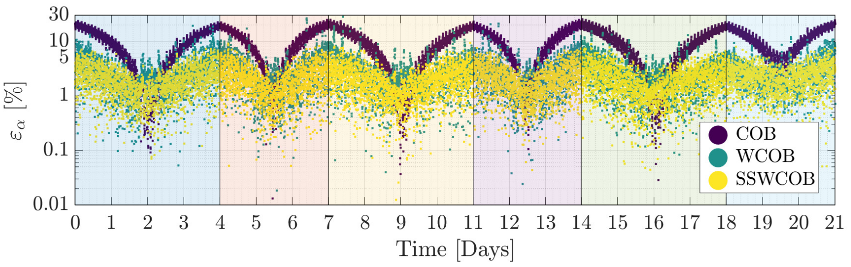
Figure 6.25: Performances of the COB, WCOB, and SSWCOB as function time during the first two arcs of the CRP. Points below *4* km are omitted for the WCOB and SSWCOB.

FRP and CRP datasets are illustrated in Figure 6.26. Note that globally, both the SSWCOB and WCOB perform better than the COB. However, as discussed previously, local spots exist that are not true. These are linked to cases at low phase angle, when D1 is fully visible. In such cases, applying the scattering law is less effective than the simple CoB. It is also noted that the performance improvement

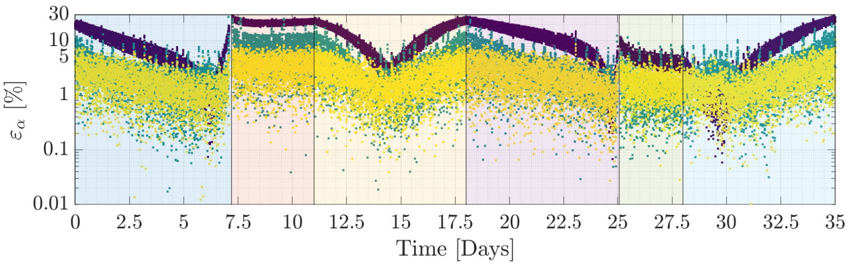
Table 6.7: Performance metrics of the IP strategies in the $\mathcal{DS}_8^{FRP}$ and $\mathcal{DS}_8^{CRP}$ datasets.

		$\mathcal{DS}_8^{FRP}$			$\mathcal{DS}_8^{CRP}$		
	Metric	COB	WCOB	SSWCOB	COB	WCOB	SSWCOB
ε_α	μ [%]	9.75	2.93	2.06	10.90	3.36	2.12
	σ [%]	6.09	2.21	1.29	7.49	3.00	1.40
ε_ψ	μ [deg]	n.a.	0.99	0.35	n.a.	1.03	0.31
	σ [deg]	n.a.	6.96	3.68	n.a.	6.40	3.67
ε_ρ	μ [m]	333.12	333.12	333.12	539.44	539.44	539.44
	σ [m]	426.33	426.33	426.33	614.65	614.65	614.65

is negligible in absolute terms, as the error on ε_α in these cases is less than *1*%.



(a) $\mathcal{DS}_8^{FRP}$



(b) $\mathcal{DS}_8^{FRP}$

Figure 6.26: ε_α for different IP strategies in the $\mathcal{DS}_8^{FRP}$ and $\mathcal{DS}_8^{CRP}$ datasets.

6.4.2 Performance of the GNC

In this section, the performance of the pointing and position reconstruction of the GNC are assessed over the $\mathcal{DS}_8$ dataset described in section A.3. For this assessment, the arc 4b of the FRP is selected as a typical example of performance assessment using the metrics defined in Section 1.8.5.

The simulations used considered closed-loop scenarios in Matlab/Simulink. Surrogate models simulate the ADCS and the actuators. The former follows the primary pointing provided by the GNC with a realistic profile that considers maximum Sun exposure for the solar panels and control of the Solar Array Drive Assembly (SADA). Inside the surrogate model, the estimated angular velocity and spacecraft attitude are affected by Gaussian noises of *0.01* deg/s and *30* arcsec, respectively, at 1σ. Similarly, the true pointing error and pointing stability are modeled perturbing the target attitude with a tuned Gauss-Markov process noise having a sigma of *46* arcsec and characteristic time of 2s. Lastly, the estimated Sun direction, being fed directly into the GNC, keeps the sensors' accuracy, namely *3.67* deg at 1σ on each axis. The initial conditions to initialize the filter are assumed to be available onboard after an uplink phase.

6.4.2.1 Pointing

The pointing performance is evaluated using different guidance strategies and IP modes to figure out the best-performing one, as illustrated in Figure 6.27.

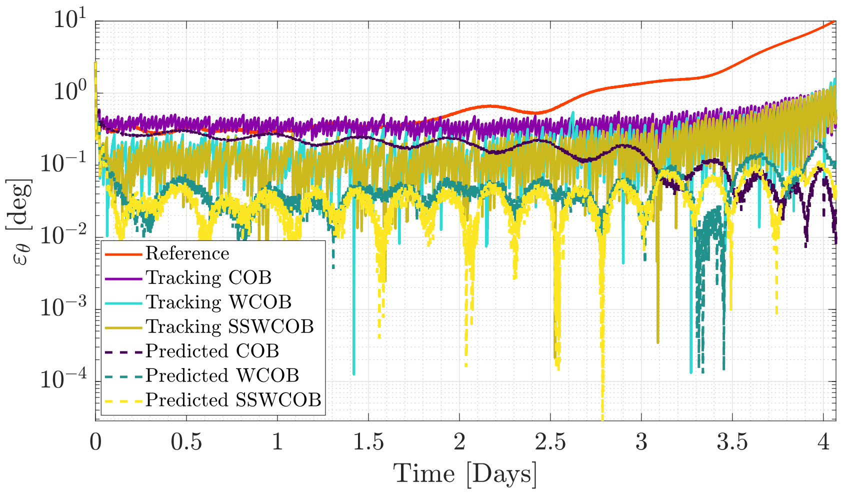


Figure 6.27: Pointing error for each combination of guidance and IP modes for one of the arcs of the FRP.

The pointing strategies using the EKF outperform all the others considered. Overall, the *Predicted WCOB* and *Predicted SSWCOB* are the ones performing best in this scenario, followed by *Tracking WCOB* and *Tracking SSWCOB*. For most of the arc, the strategies based on the COB algorithm give significantly worse results than the ones based on other IP techniques. The error using the *Reference* strategy increases considerably over time. This is caused by the accuracy of the onboard ephemerides, which degrades as time goes by because of uncertainties. From a pointing perspective, if *Reference* is used as a nominal strategy for the pointing, D1 could be lost by the navcam FOV towards the end of the arc, coinciding with the CubeSat getting closer to the system.

Towards the end of the arc, the pointing error increases for all strategies considered but the COB-based ones. This was expected from the static performance assessment in the previous section since, in these cases, D1 will be imaged at close distance and low phase angles, the latter being an optimal condition for the COB.

6.4.2.2 Position estimation

The onboard position reconstruction error $\varepsilon_{\mathbf{p}}$ is illustrated in Figure 6.28. Three different phases are identified for the strategies using the EKF. At first, after initialization, the error rapidly decreases at the beginning of the arc, showing quick convergence of the EKF. This phase is followed by one in which the errors display a steady but constant increase and terminate by a phase in which a rapid drop is observed towards the end of the arc. On the other hand, the error with the *Reference* strategy, using the ephemerides stored onboard, shows a steady but constant increase throughout the arc.

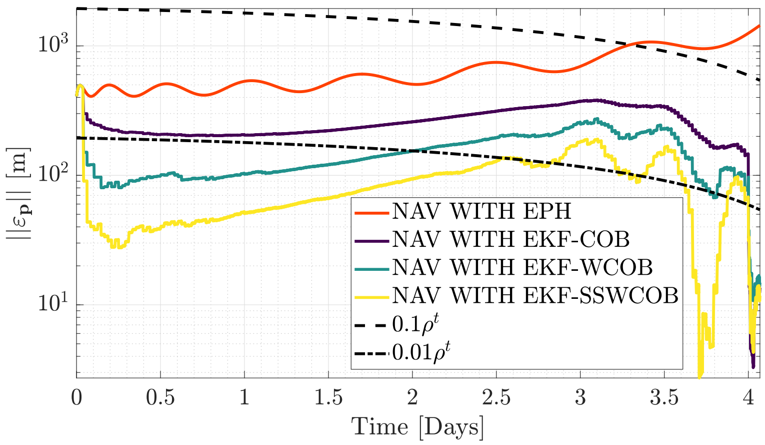


Figure 6.28: Position estimation error for each combination of navigation and IP modes for one of the arcs of the FRP. The curve of *10%* and *1%* of the true range ρ^t from D1 are represented by black dashed lines. The *10%* curve represents the target requirement for the performance of the onboard navigation.

The accuracy of the IP method used consistently affects the performance of the EKF, which in all cases is considered to perform better than the reference scenario. Lastly, it is also noted that the different trends between the EKF error in Figure 6.27 and the *Predicted* one in Figure 6.28 is attributed by the error having a large component in the boresight direction. Since this does not actively contribute to the pointing error, different trends are observed between ε_θ and ε_p.

6.5 Towards flight operations

Towards the end of the design phase, to prepare the IP for flight operations, incremental tests have been executed to validate its robustness and functioning.

This section illustrates these tests, showcasing the stress they put on the IP pipeline and phenomena to be expected during flight operations.

The assessments are focused on the IP, since, as illustrated in the previous section, it is the main driver in the performance of the GNC.

6.5.1 Images-in-the-loop

An image-in-the-loop campaign has been performed in [119] exploiting the TinyV3RSE facility as HIL setup to mimic the acquisition of real-world, noisy images. The facility is setup with a representative navcam (the default Basler camera in TinyV3RSE, see Section 3.2) to generate the datasets $\mathcal{DS}_7$ illustrated in Section A.3.

First, images are rendered in CORTO varying essential parameters such as the relative positions of D1, D2, Milani, and the asteroids's shapes. This provides a vast dataset with a comprehensive set of possible geometrical configurations and uncertain-before-arrival asteroid parameters. Second, images are acquired in TinyV3RSE with diverse exposure time and blur to understand the behavior of the IP in these untested settings. Figure 6.29 and Figure 6.30 display a sample of facility images acquired with different exposure times and blur.

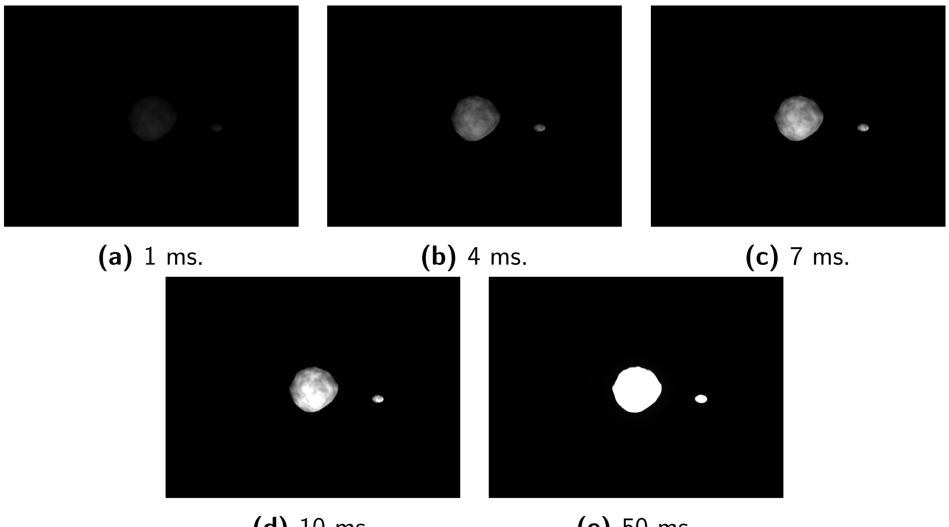
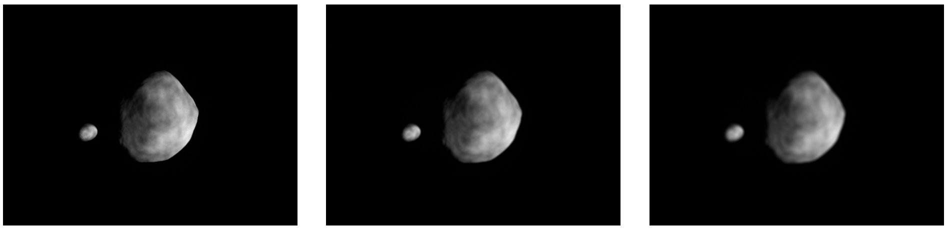

(a) 1 ms. **(b)** 4 ms. **(c)** 7 ms. **(d)** 10 ms. **(e)** 50 ms.

Figure 6.29: Facility images with different exposure times.

The testing campaign focused only on the performance of the WCOB, since it is the most complex from an algorithmic point of view and since it includes steps that are also shared with SSWCOB and COB, making it possible to generalize performance.

Synthetic images reproduced in CORTO are projected into the TinyV3RSE screen, acquired by the camera, and then corrected to compensate for facility errors estimated during TinyV3RSE's calibration. Whenever possible, the IP is run for



(a) $\sigma_{\text{blur}} = 0.77$ pixel. **(b)** $\sigma_{\text{blur}} = 1.47$ pixel. **(c)** $\sigma_{\text{blur}} = 2.60$ pixel.

Figure 6.30: Facility images with different blur levels.

each dataset on its facility and synthetic version. This allows to properly evaluate the impact of the hardware on substantially geometrically equivalent images.

The main results for the different test cases of $\mathcal{DS}_7$ are discussed below, while a summary of the performance metric statistics is illustrated in Table 6.8, while in [119] it is possible to consult a comprehensive set of histograms and boxplots detailing the performance for each dataset.

Table 6.8: Summary of the results. The mean μ and standard deviation σ of the estimation errors are reported for each test case. The nomenclature used for the test cases is: ${}^{\text{ScaledAxes}}\text{Dataset}^{\sigma_{\text{blur}}[\text{px}]}_{\text{ExpTime [ms]}}$.

Test case	ε_u [px] $\mu(\sigma)$	ε_v [px] $\mu(\sigma)$	ε^{n}_{CoF} [px] $\mu(\sigma)$	ε_ψ [deg] $\mu(\sigma)$	ε_ρ [m] $\mu(\sigma)$
$\text{DS1}^{0.77}_{1}$	-1.13 (20.3)	-8.57 (23.2)	22.2 (23.1)	9.18 (6.92)	621.9 (522.2)
$\text{DS1}^{0.77}_{4}$	-1.06 (19.1)	-8.54 (22.2)	21.2 (22.0)	9.14 (6.84)	614.2 (525.6)
$\text{DS1}^{0.77}_{7}$	-1.34 (18.8)	-7.34 (21.5)	20.6 (21.2)	8.97 (6.86)	611.8 (527.6)
$\text{DS1}^{0.77}_{10}$	-1.03 (18.1)	-8.03 (21.6)	20.4 (21.0)	8.93 (6.75)	598.1 (520.4)
$\text{DS1}^{0.77}_{50}$	0.07 (7.27)	-4.85 (13.0)	10.7 (11.4)	-0.56 (6.45)	97.3 (304.0)
$\text{DS1}^{1.47}_{7}$	-0.08 (17.6)	-8.39 (18.5)	19.5 (18.5)	8.30 (6.85)	617.3 (525.9)
$\text{DS1}^{2.60}_{7}$	1.91 (17.9)	-8.74 (18.3)	20.0 (18.4)	7.18 (6.90)	651.1 (532.9)
${}^{\text{xyz}}\text{DS1}^{0.77}_{7}$	0.64 (15.6)	-4.86 (18.1)	17.5 (17.0)	8.94 (6.86)	1096 (642.7)
${}^{\text{xy}}\text{DS1}^{0.77}_{7}$	0.86 (13.2)	-5.36 (20.5)	17.4 (17.9)	10.4 (7.13)	751.1 (539.1)
${}^{\text{z}}\text{DS1}^{0.77}_{7}$	0.42 (19.3)	-4.17 (17.2)	19.6 (17.4)	8.39 (8.62)	894.9 (672.4)
$\text{DS2}^{0.77}_{7}$	-1.41 (16.5)	-6.19 (22.2)	20.5 (19.7)	9.95 (6.64)	471.6 (495.2)
$\text{FRP}^{0.77}_{7}$	-0.28 (6.51)	-2.59 (7.68)	8.78 (5.56)	9.82 (6.44)	604.0 (483.2)
$\text{CRP}^{0.77}_{7}$	5.15 (8.23)	-1.19 (12.2)	12.5 (9.36)	10.3 (6.08)	420.5 (368.6)

The impact on the algorithm performance is minimal for short exposure times, from *1*ms to *10*ms in the TinyV3RSE setup used. Increasing the exposure time causes the performance of the facility images to be similar to that of synthetic images. The low impact of the exposure times is attributed to the binarization procedure executed at the beginning of the IP pipeline. The binarization step performs consistently and robustly at varying exposure times, given that the threshold is autonomously selected onboard using Otsu's method [106].

From Table 6.8 it is possible to observe that the *u* and *v* components of the CoF are generally estimated well, as the median of the error is close to *0*, while a significant bias is visible for Ψ and ρ. The binarization step again explains this: facility images show a higher intensity variability with respect to synthetic images, and therefore, a higher binarization threshold is usually selected using Otsu's method. As a result, more pixels are cut out from the binary image, resulting in a smaller blob of pixels associated with D1 than the synthetic one considered during training of the fitting coefficients. This phenomenon is pivotal in explaining the bias that is introduced in the range and phase angle, which are directly dependent on the characteristics of this blob. In particular, the range is determined from the blob's semi-major axis. Therefore, a smaller blob leads to an overestimation of the range. The phase angle, instead, is estimated from its eccentricity. Since the additional pixels that are cut out from the facility images are usually grouped around the terminator region of the asteroid, the eccentricity of the binarized blob tends to increase, resulting in a higher estimate of the phase angle, as per Equation 6.3.

These effects are appreciable in Figure 6.31, where the difference between a facility-binarized image and a synthetic-binarized one is shown, together with the CoB of the two blobs and the corresponding fitted ellipses. The gray area is the one that is retained in the binarized synthetic image but discarded in the binarized facility image. As explained before, this leads to a shift of the CoB (the blue/red dot) away from the CoM and to a fit with an ellipse with different properties. The blue ellipse, obtained from the fit of the binarized facility image, is characterized by a smaller semi-major axis and a higher eccentricity with respect to the synthetic one.

From this analysis, it can be concluded that the primary mechanism affecting the performance of the IP is driven by a difference in the geometrical properties of the blob of pixel of D1 caused by the binarization algorithm acting differently at different exposure conditions. In general, this phenomenon causes biases in the performance of the IP whenever facility images are used and caution towards the proper tuning between the properties of the datasets used during training and the setup that will be used during the mission for acquisition.

Concerning the different blur levels, the IP proved robust, showing minimal to no impact on performance caused by increased blur levels. The only appreciable trend is in the error on the *u* component of the CoF, which tends to spread more as the blur level increases.

A variety of phenomena has been observed concerning the different D1 scaling across *xyz*, *xy*, and *z* axes. These are varied with $\pm 5\%$ of their nominal values from [138].

Considering CoF performance, the WCOB improves when scaling all the body axes and the *xy* axes. This can be explained by considering that for a smaller body, the CoB will be closer to the CoM, which reduces the need for corrections. Indeed, when only the *xy* axes of the body are scaled, the *u* component of the CoF error is smaller, while the *v* component behaves similarly to the nominal case. Recall that,

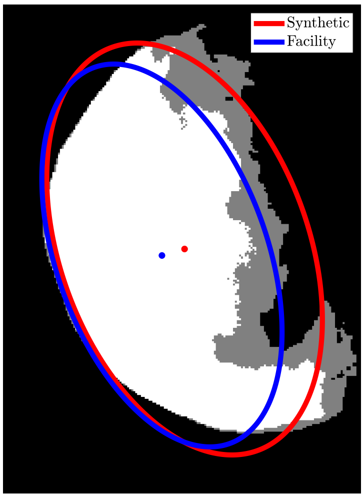


Figure 6.31: Comparison between the blobs of pixels obtained from the binarization of a facility and synthetic image. The fitted ellipse and CoB resulting from the blob analysis are also shown.

since in the $\mathcal{DS}_7$ dataset the spacecraft elevation is limited between ± 45 deg and no boresight rotations are implemented, changes in the x and y axes of the body reflect mainly on the u direction of the image, while changes in the z axis of the body reflect mainly on the v direction of the image. Indeed, when scaling the z axis, the performance in the u direction is worse than the nominal case, while the error in the v direction is smaller.

Considering the phase angle, the only observed main effect has been a slightly wider distribution spread in the performance when considering scaling of the z axis. The range estimation, however, results significantly impacted by the shape change, especially when considering changes on the z axis. This is expected as the scaling will make the body appear considerably smaller in the image, introducing a bias on the semi-major axis used in Equation 6.1 to compute the range from D1. Finally, it is commented that scaling has coherent effects both on synthetic and facility images.

Finally, the performance is assessed for the datasets representing nominal orbits in Milani, exhibiting the same trends discussed above.

Testing the WCOB under various conditions by considering different geometrical configurations and hardware settings with an equivalent camera model of the one that will be used on Milani gives more confidence about the robustness and reliability of the algorithm for flight operations. The results show the algorithm's robustness in all the considered test cases. Indeed, the algorithm can provide good estimates of the CoF coordinates, even in challenging conditions. Instead, the ψ and ρ estimates are more sensitive to hardware effects and have shown biased results. Nevertheless, they are of secondary interest outside of the IP since they are just an internal byproduct and are not currently used directly for navigation purposes. However, the bias introduced by the Ψ is causing degraded performance in the CoF estimate, which is reflected by the greater variance of the CoF error distribution with facility images. All these effects can be considered by specific tuning via datasets made of images more similar to the ones that will be encountered during nominal flight conditions and with the acquisition setup that will be used during flight. Also, the possibility of using TinyV3RSE-generated images can be investigated to close the domain gap between acquisition and design datasets.

6.5.2 Hardware-in-the-loop

A qualitative assessment has also been performed with an engineering model of Milani's navcam. The purpose of this activity has been to test IP performance with the engineering model of the navigation camera to test the robustness of the IP to the proper characteristic noise of the camera and different illumination conditions on the asteroid.

An image of the engineering model of the navigation camera mounted within the TinyV3RSE facility is illustrated in Figure 6.32. A 3D-printed support has been designed specifically for the camera to allow easy mechanical interface in the facility. An example of the image projected on the screen in TinyV3RSE and the image acquired from the camera is illustrated in Figure 6.33. Note that due to the unavailability of the proper collimator lens, geometric equivalence was not reached during this test, as the image captured appeared bigger than it should have. For this reason, the test represents only qualitative assessment. Nonetheless, it was fundamental to characterize for the first time the interfacing with the navigation camera, its noise characteristics, and the photometric response of the images that could be acquired during flight operations, the latter being an essential aspect of the performance of the IP, as illustrated in Section 6.5.1.

It is also commented that during the design process of the mission, the sensor properties changed from a 2048×1536 px grayscale sensor to an RGB one of equivalent resolution. A performance assessment has been conducted to test the functionalities of the different IP strategies to work with images after the application of Bayer filtering, but without applying interpolation or demosaicing to keep the computational cost of the algorithm low. The IP demonstrated adequate flexibility in this case, which has also been tested during the HIL qualitative assessment. This was possible given the expected albedo variations on the body and the natural color of the body (grayish) that would activate the $RGGB$ pattern of the Bayer's filter mostly homogeneously.

6.5.3 Training and testing during flight

As part of the nominal objectives of the mission, Milani will conduct an Autonomous Optical Navigation experiment (AutOpNav). AutOpNav is an opportunistic tech-

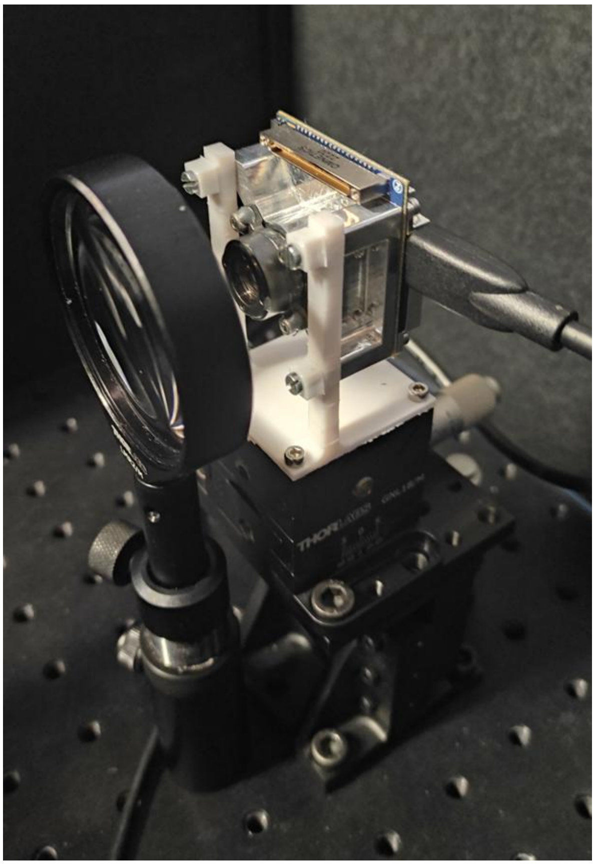
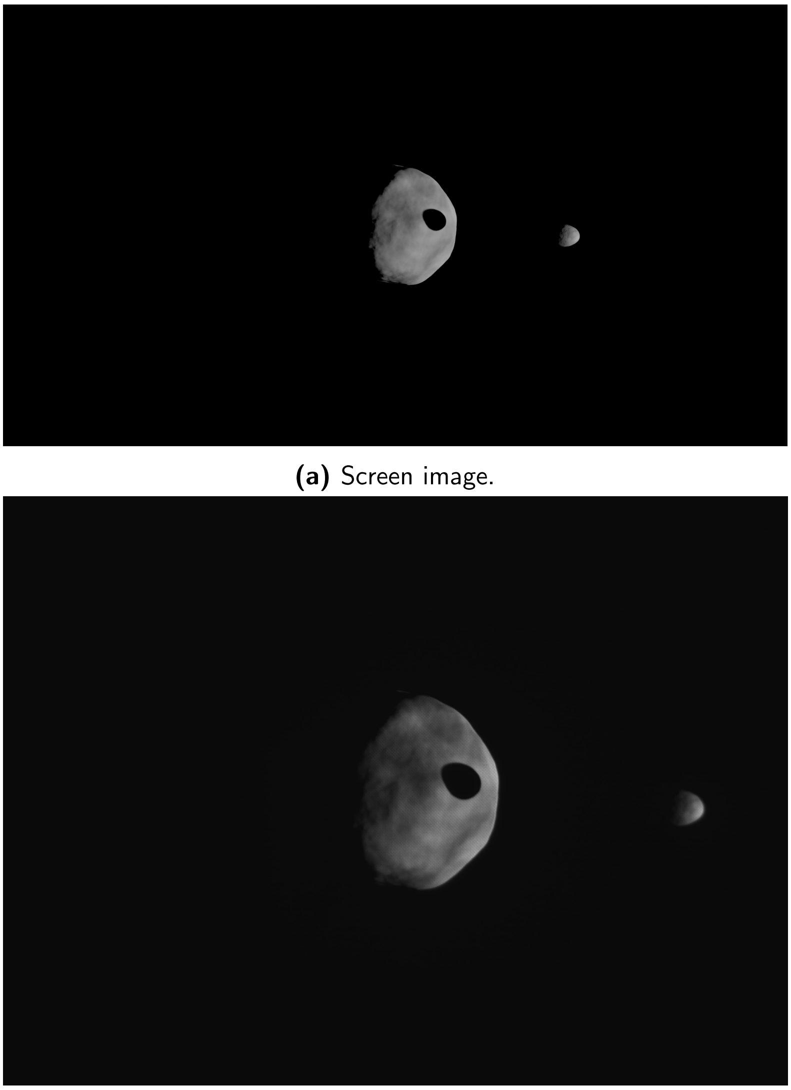


Figure 6.32: Engineering model of the navcam of Milani mounted within the TinyV3RSE facility.

nology demonstration experiment whose goal is to characterize Milani's optical navigation during flight and its capability to support autonomous navigation in close proximity of the Didymos system.

To do so, an on-ground infrastructure is proposed to compare and replicate the outcome of the onboard algorithms. The concept of the experiment is illustrated in Figure 6.34. Its core consists of the on-ground asynchronous and opportunistic collection of the onboard data from the IP, navigation algorithms, and sensors. On the ground, the data is used to demonstrate the performance of the onboard algorithms. To validate the filter's performance, the Milani trajectory's ground-based solution is used as a reference. Then, a fictitious trajectory maneuver is computed based on the onboard navigation filter's solution, compared with the one produced by the flight dynamics team. In this way, the experiment also evaluates the accuracy obtained with a fully autonomous guidance cycle.

The IP performance can be validated using the optical navigation images that are downlinked to perform the nominal Orbit Determination (OD) process for the CubeSat. No additional images need to be downlinked for the experiment. To


(a) Screen image.

(b) Facility image.

Figure 6.33: Example of an image of the Didymos system projected on the screen in TinyV3RSE (a) and acquired by the engineering model (b).

avoid further burden on the data volume to be downlinked, real images used for the OD will be complemented with synthetically generated ones rendered at the reconstructed poses of the CubeSat to simulate the correct activation frequency of the onboard IP. These images can be used directly or as seen in TinyV3RSE, where an engineering model of Milani's NavCam could be used.

Setting up the infrastructure for the execution of the experiment will open up a set of possible related activities that could benefit the mission and widen the scope of the experiment. For example, real data from the spacecraft could be used to test the performance of alternative IP and navigation algorithms. Data from

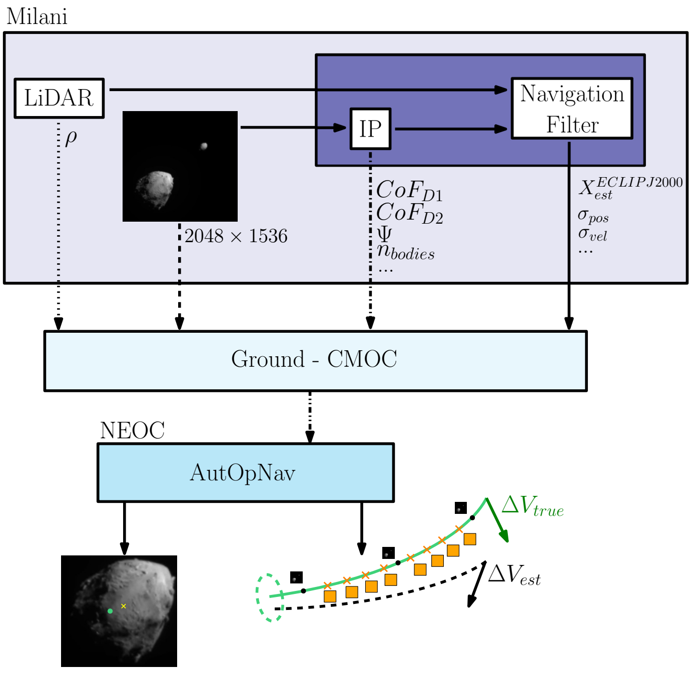


Figure 6.34: Proposed high-level concept of operations for the AutOpNav experiment.

DART and Hera could also be used before Milani's release or during its mission to tune the parameters of the data-driven algorithms. The former option would provide valuable initial insight into the performance of Milani's onboard algorithms with actual images of Didymos.

The AutOpNav experiment is designed to be carried out at the Navigation Experiment Operation Centre (NEOC), where the AutOpNav experiment team will retrieve the spacecraft telemetry and ground operation products from the CubeSat Mission Operation Centre (CMOC) Supervision Level. The team will be a user of such data, similar to the science teams, without interfering with the operations of Hera.

The NEOC will acquire and store the incoming data from CMOC and process them in a dedicated processing station. Different functional levels are envisioned. At the highest level, telemetry data about the reconstructed state of the onboard filter and the IP output will be compared to their reconstructed ground truth to generate error metrics, which will validate navigation performance and achieve. Additional levels are devised that complement available data with artificial ones generated in a dedicated rendering station and/or obtained in the TinyV3RSE facility. The preliminary functional architecture of the NEOC is reported in Figure 6.35.

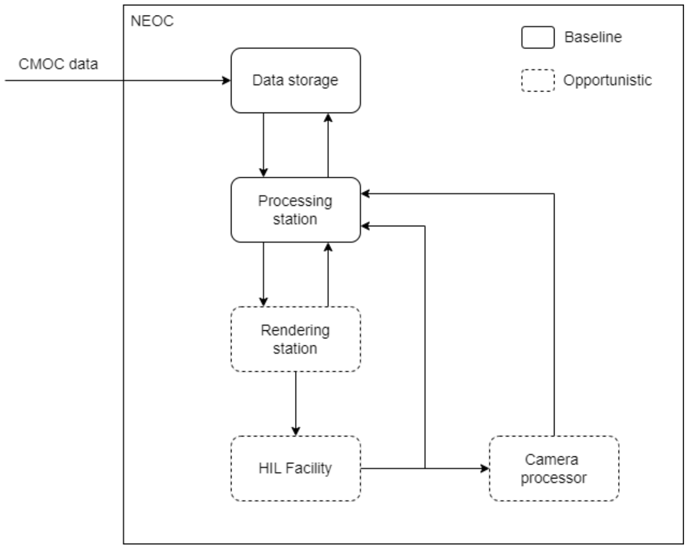


Figure 6.35: Proposed high-level architecture of the NEOC.

6.6 AI enhancement

Over the course of the research activity performed during the Ph.D., the Milani mission and the Didymos binary system have always been considered the natural case study. This section summarizes the most attractive applications of data-driven and AI approaches that could benefit the Milani mission. The nature of the baseline IP of Milani resonates positively with AI methods, allowing for simple and direct comparisons.

6.6.1 Comparison with NN and CNN

Adopting the same methodology used to design the data-driven IP of Milani, NN and CNN architectures are investigated, presenting the work in [32]. The scope is to understand what kind of performance gain would be possible by implementing other data-driven methods using the same input of the baseline algorithm for the mission, represented by the COB and WCOB. $\mathcal{DS}_7^{DB_0}$ is used for training and validation, while the performance of AI-based methods are assessed on test sets represented by the $\mathcal{DS}_7^{FRP}$ and $\mathcal{DS}_7^{CRP}$ datasets.

While WCOB, NN, and CNN are all data-driven methods, they exhibit substantial differences. Both the WCOB and NN work on a set of explicitly selected features extracted from images with traditional techniques. While the WCOB acts on these features using explicit hand-crafted relationships fitted through data, the NN approach uses implicit relationships embedded in the network and learned through the data. On the other hand, CNNs do not use as input the same features vector as WCOB and NNs, but instead, learn a representation of the feature vector (in the form of the fully connected layer) via its convolutional layers adjusted during training.

This reflects a different usage of the data of $\mathcal{DS}_7$ by different methods. In the WCOB, a scattering law is empirically derived from data and applied to the CoB estimate. The data is needed to tune the fitting coefficients of a few core functions used in the scattering law, while most of the functionalities in the pipeline retain a traditional approach. Similarly, both NN and CNN need data, which is used more efficiently to optimize their internal structure, namely to find optimal sets of weights and biases.

The NN architecture is designed to use an input the feature vector illustrated in Section A.3.1, while the CNN uses cropped and resized versions of the images acquired by the navigation camera. A *80/20%* partition of the $\mathcal{DS}_7^{DB_0}$ trains and validates the AI methods.

As it is possible to see in [32], different architectures have been trained and tested. However, to simplify the discussion, only the most relevant ones are reported in this section. These are the NN represented in Table 6.9 and the CNN represented in Table 6.10.

Table 6.9: Architecture of the NN. The total number of parameters is *21,347*(0.08 MB), all trainable.

ID	Layer type	Output Shape	Parameters	Connections
I	Input	($\mathcal{B}$, 14)	0	D1
D1	Dense	($\mathcal{B}$, 32)	480	D2
D2	Dense	($\mathcal{B}$, 64)	2112	D3
D3	Dense	($\mathcal{B}$, 128)	8320	DO
DO	Dropout	($\mathcal{B}$, 128)	0	D4
D4	Dense	($\mathcal{B}$, 64)	8256	D5
D5	Dense	($\mathcal{B}$, 32)	2080	D6
O	Dense	($\mathcal{B}$, 3)	99	

Both the NN and CNN have been trained using *Adam* optimizer, and the MSE as error metric with a batch size of *32* samples, *ReLu/Sigmoid* as activation functions. The NN has been trained for *500* epochs with a learning rate of *0.02*, while the CNN has been trained over *60* epochs with a learning rate of *0.05*. More details on the training can be seen in [32].

6.6.1.1 Results

Figure 6.36 summarizes the performance of the different methods over the testing dataset $\mathcal{DS}_7^{FRP}$ using the same familiar plots used to illustrate the performance of the IP of Milani in Section 6.4.1. The performance on $\mathcal{DS}_7^{CRP}$ is omitted for clarity but can be consulted in [32].

From Figure 6.36, it is possible to see that WCOB and NN achieve similar performance, better than the COB method, while the CNN outperforms all the methods considered. A NN operating with a similar set of inputs than the ones of the WCOB does not improve the performance in a relevant way. The WCOB method

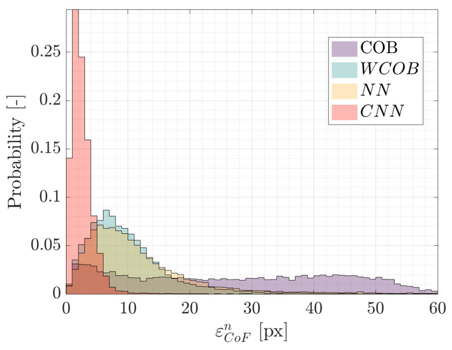
Table 6.10: Architecture of the CNN. The total number of parameters is *1,438,659* (5.5 MB), all trainable.

ID	Layer type	Output Shape	Parameters	Connections
I	Input	($\mathcal{B}$, 128, 128, 1)	0	C1
C1	Conv2D	($\mathcal{B}$, 128, 128, 32)	320	P1
P1	MaxPooling2D	($\mathcal{B}$, 64, 64, 32)	0	C2
C2	Conv2D	($\mathcal{B}$, 64, 64, 64)	18496	P2
P2	MaxPooling2D	($\mathcal{B}$, 32, 32, 64)	0	C3
C3	Conv2D	($\mathcal{B}$, 32, 32, 128)	73856	P3
P3	MaxPooling2D	($\mathcal{B}$, 16, 16, 128)	0	C4
C4	Conv2D	($\mathcal{B}$, 16, 16, 256)	295168	P4
P4	MaxPooling2D	($\mathcal{B}$, 12, 12, 256)	0	FC
FC	Flatten	($\mathcal{B}$, 16384)	0	D1
D1	Dense	($\mathcal{B}$, 64)	1048640	DO
DO	Dropout	($\mathcal{B}$, 64)	0	D2
D2	Dense	($\mathcal{B}$, 32)	2080	O
O	Dense	($\mathcal{B}$, 3)	99	

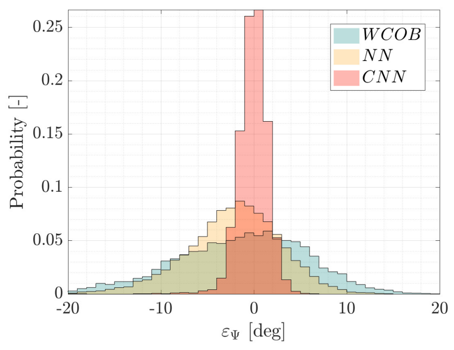
being fully explainable means that it can be deployed with much more confidence than a NN on a flying mission because its main blocks are well-understood, robust, and known in the literature. On the other hand, if much higher accuracy is sought, the CNN can outperform all other methods considered.

Globally, the COB is the best for only *3.40%* of the images and mostly at lower phase angles. The WCOB and NN are the best in the low-to-medium and medium-to-high intervals of phase angles, ranking best respectively in *7.30%* and *5.14%* of the cases. Finally, the CNN is considered the best-performing method across various conditions. *84.17%* of the images in $\mathcal{DS}_7^{FRP}$ is performed with the smallest ε^n_{CoF} error with this method.

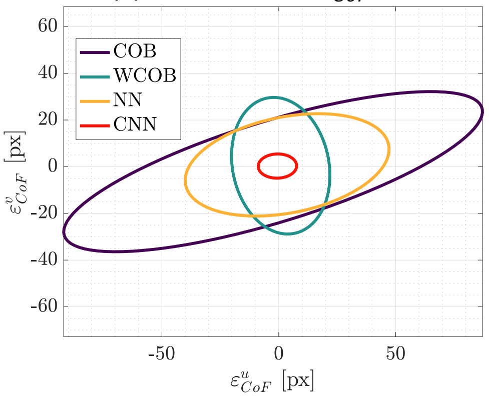
The difference in performances between WCOB, NN, and CNN are explained by the type of input and methods adopted. The NN approach works with explicit features determined from traditional IP, the WCOB uses a subset of these and one filter, while the CNN uses a large set of filters and an implicit features vector which are not defined a-priori but determined through training. The NN proves that the information extracted from filters is optional to achieve good performance. A simple feature vector is enough to achieve the same performance of the WCOB, meaning that the intricate non-linear relationship between input and output that can be established with a NN could easily substitute the various steps required by the hand-crafted WCOB pipeline. On the other hand, the CNN demonstrates a boost in performance that hints that extracting spatial information from the image is a key functionality in generating a better implicit feature vector. In between, the WCOB method proves that traditional IP functions, the application of a single filter, and a few explicit features extracted from the image are enough to obtain good results in the contexts of a complex, hand-crafted IP pipeline.



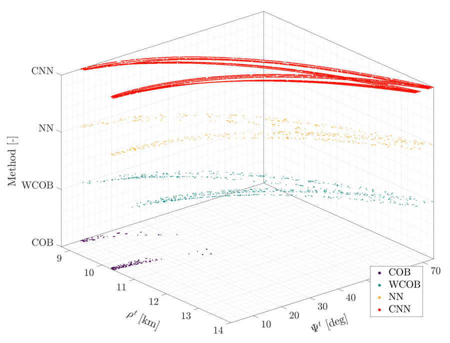
(a) Histograms of ε^n_{CoF}.



(b) Histograms of ε_ψ.



(c) Error ellipse.



(d) Best methods.

Figure 6.36: Histograms of the ε^n_{CoF} (a) and ε_ψ (b) of different methods. The binwidth is set to *1* pixel and *1* deg, and *probability* is used as normalization. (c) Error ellipse in the image plane of ε^u_{CoF} and ε^v_{CoF} with a *99%* confidence interval. The points are omitted for clarity. (d) Scatter plot of the method with the smallest ε^n_{CoF} as a function of the range and phase angle. All the plots are illustrated for $\mathcal{DS}^{FRP}_7$.

6.6.1.2 Explainability analysis

The similarity of the performance between NN and WCOB has motivated the explainability analysis illustrated below. This analysis is conducted to understand which optical observables can be generated by the blob analysis within the *Blobs Characterization* block are meaningful to generate the desired output. This is useful for a twofold purpose: to better explain the underlying implicit pipeline of the NN method and to understand if the proper set of parameters has been identified during the design of the WCOB.

The explainability analysis is performed using SHAP values [159], which are briefly explained. These implement a game theory approach that breaks down the contribution given by each player to the results of a game. Analogously for the case considered in this section, SHAP values quantify the contribution of each feature on the model prediction, independently from its complexity. They follow a formulation and statistical properties, which are discussed in detail in [159]. In a nutshell, SHAP values give an interpretation of the impact of a given feature, exploring all the possible model predictions generated by a coalition between such a feature and the remaining ones. However, doing so for all possible combinations would not be tractable, so approximations and samplings are necessary, as explained in [159]. The SHAP values can then be used as a proxy to visualize the impact of a given feature on the model prediction. The higher the SHAP value in magnitude for a particular feature, the higher its effect on the output.

The SHAP values of a sample of 2500 random cases from $\mathcal{DS}_7^{DB_0}$ of the NN predictions are computed with a kernel explainer. From Figure 6.37 it is possible to see from a global perspective the impact of each feature by looking at the mean of the absolute SHAP values.

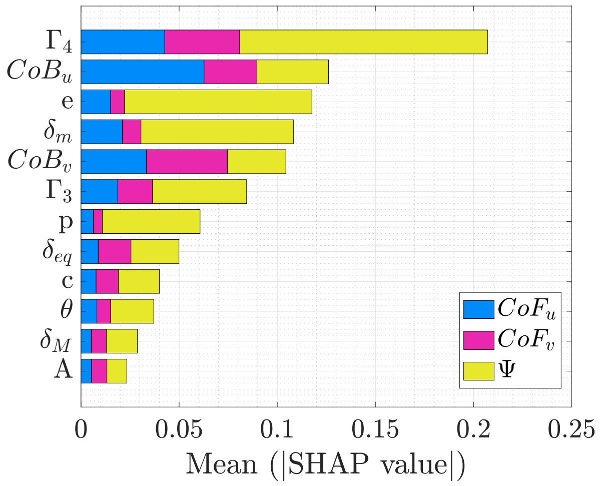
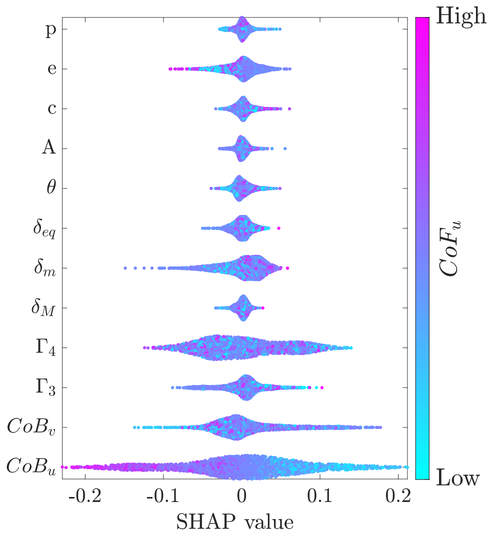
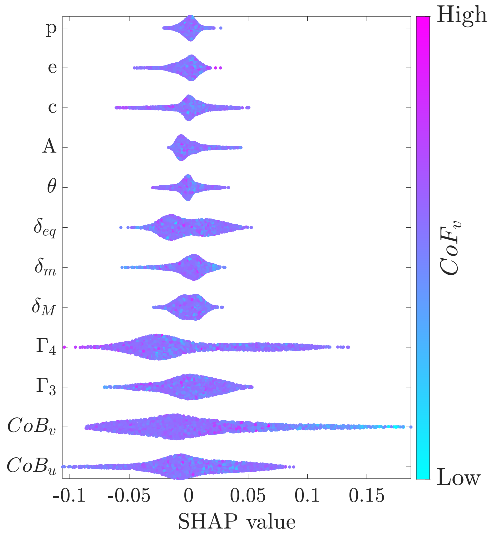


Figure 6.37: Stacked histogram of the mean absolute SHAP values of the NN output for each element of the input feature vector.

Ψ is mostly driven by Γ_4, e, and δ_m. Of these features, only the eccentricity e is range invariant. In contrast to the WCOB, the NN seems capable of synthesizing a more accurate Ψ estimate when combining the eccentricity with the height of the bounding box and the minor axis length. This hints that it would be possible to slightly improve the Ψ estimate more than that of the WCOB when considering multiple optical observables. Such a new formulation's complexity must be balanced against the expected improvement. On the other hand, the CoF estimate is driven in the u component by the CoB_u and Γ_4 while in the v component by CoB_v and Γ_4. This hints at a possible relationship found by the NN between CoF and CoB and Γ_4 components. The NN may have learned to correlate a correction between CoF and CoB and to scale it properly as a function of the range from the body by exploiting Γ_4 as a proxy for the range. Also, this case reflects a possible change in the features selected in the WCOB method. Alternatively, the SHAP values are illustrated case by case for the three outputs of the NN_2. In Figure 6.38, it is interesting to observe the correlation effects of combinations of low-high features and low-high SHAP values. For example, it is possible to observe how the CoF_u and CoF_v are largely influenced by the CoB_u and CoB_v, respectively. However, The former shows polarization between SHAP values and feature values: low values of the CoB_u tend to output low values of CoF and vice-versa. However, the same is not true for the CoB_v, which makes sense since the geometry of the problem (and consequently the Ψ) is affecting the estimate mostly around the CoF_u component, given how the dataset $\mathcal{DS}_7$ is generated. Another interesting effect of taking note of is the behavior of Γ_3 and Γ_4 for the CoF: in both components, high values of Γ_3 are related to low values of the CoF components, while high values of Γ_4 are related with high values of the CoF components.



(a) SHAP values for CoF_u.



(b) SHAP values for CoF_v.

Figure 6.38: SHAP values of the input features of the NN method for the CoF_u (a) and CoF_v (b) outputs, colored by the feature's value.

Finally, the SHAP values are analyzed for the Ψ estimate in Figure 6.39. It is very interesting to observe that the major contribution to the output is in order γ_4, eccentricity, length of the minor axis, and perimeter (as seen in Figure 6.37). These parameters show a strong polarization between feature values and SHAP values. Of these quantities, the eccentricity is the only range invariant. High eccentricity values correspond to high values of phase angle and vice-versa, the same relationship exploited in the WCOB method (see Figure 6.10). It is then interesting to observe that also γ_4, the length of the minor axis and perimeter could be exploited efficiently in the phase angle estimate, as high values of phase angle are correlated with low values of perimeter and minor axis length and higher values of γ_4 and vice-versa. All these quantities can be considered a proxy of the eccentricity, although they are not range-dependent.

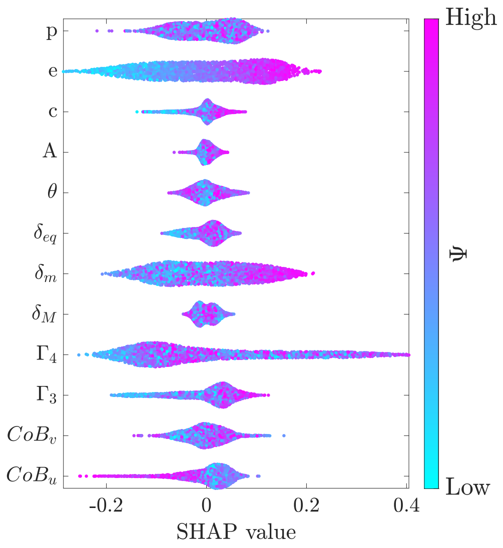


Figure 6.39: SHAP values of the input features of the NN method for the Ψ output, colored by the feature's value.

6.6.2 Navigation with segmentation maps and CNN

This section briefly illustrates the results of the approach presented in Section 5.2 for a close-proximity scenario about Didymos represented by the $\mathcal{DS}_5^{D-4}$ dataset. These results demonstrate how a deep-learning architecture performing classification can be used for navigation using segmentation maps as input.

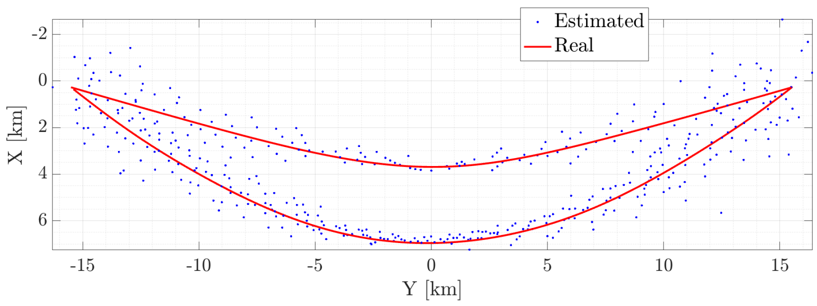
Dataset $\mathcal{DS}_5^{D-4}$ is made by 432 samples of Didymos imaged with a 1024×1024, 16×16 deg FOV sensor from a portion of the Milani's operational orbit. Images are taken every 10 minutes with ideal pointing in a 3-day loop for 432 image-mask pairs.

The results of classification portion of the architecture described in Section 5.2 are represented in polar coordinates in Figure 6.41, while the estimated positions are superimposed on the actual trajectory in Figure 6.40.

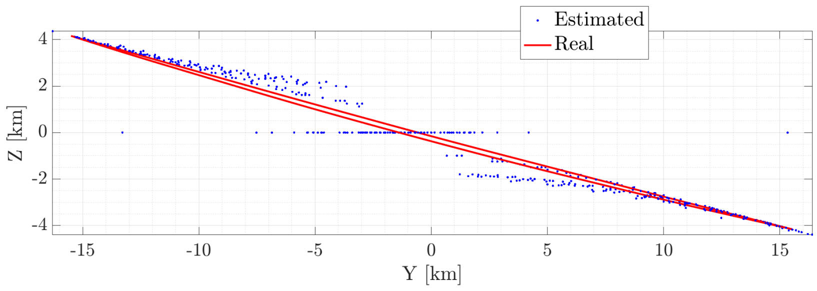
A relevant portion of the trajectory happens outside the $[0.65 - 1.35]D_0$ interval considered during training. Whenever the range exceeds this interval's upper limit, the images are scaled according to the postprocessing presented in Section 3.1.1.5. The proposed architecture performs well in predicting the equatorial angle θ and range ρ but fails to accurately predict the azimuth angle ϕ. This is due to the classes chosen for this variable, which turned out to be too broad (developing around ± 15 deg) for the target orbits considered.

As it is possible to see in Figure 6.41 and Figure 6.40, the reconstructed trajectory exhibits significant errors. This is caused by the nature of the CNN architecture, designed to predict the position as a classification task, generating a coarse estimate. For completeness, the error of the reconstructed trajectory is illustrated in Figure 6.42.

From Figure 6.42(a), it is possible to see a high relative percentage error



(a) XY view.



(b) YZ view.

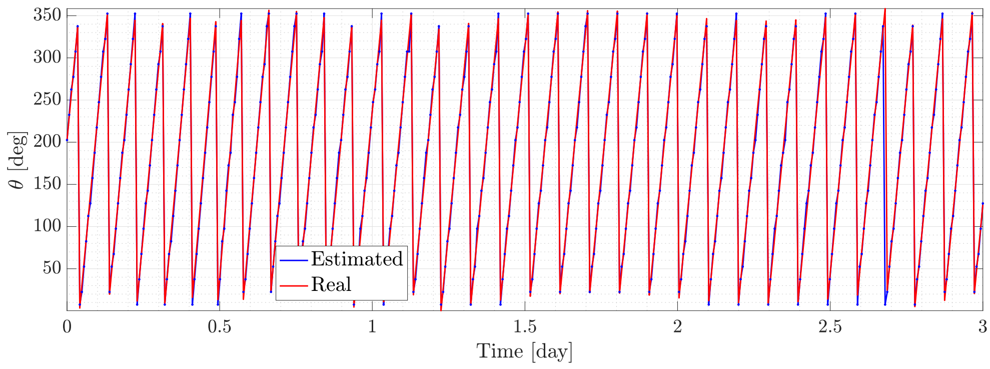
Figure 6.40: Close-proximity orbit example around Didymos (red) and estimated positions with the CNN (blue). The trajectory is illustrated in the $\mathcal{W}$ reference frame, with the X axis pointing towards the Sun.

with respect to the true range $\epsilon^r_{\mathbf{p}}$ as a function of the range. The mean relative percentage error is 9.92%. Comparing the performance illustrated in Figure 6.42(a) of the classification portion with the one in Figure 5.17(a) which includes the NCC portion, is possible to see the improvement generated by the combined methodology.

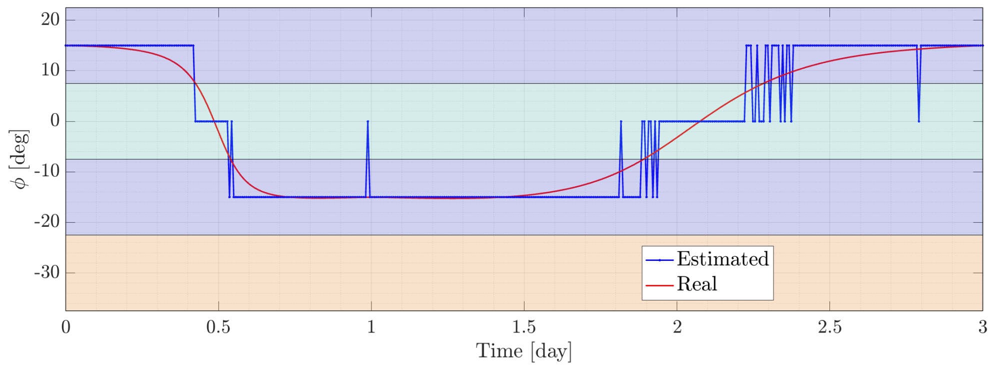
Finally, from Figure 6.42, it is possible to see that the high error is caused by significant contributions in the X and Y components in the $\mathcal{CAM}$ reference frame. Moreover, the error on the Z axis is believed to be lower due to the postprocessing strategy illustrated in Section 3.1.1.5, which artificially augments the number of classes for inference, increasing their resolution and thus the performance of the estimation in the boresight direction.

6.6.3 Testing the IP after DART updates

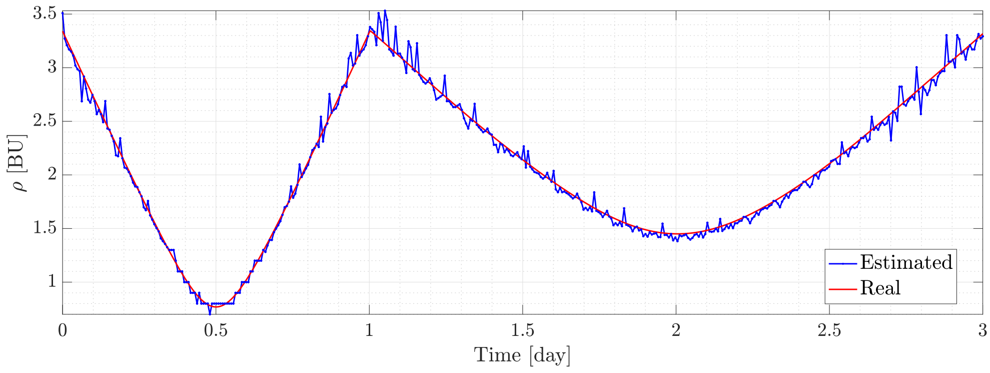
By design, Milani's IP has been structured as a pipeline that uses coefficients to express data-driven scattering laws. These coefficients can be changed throughout different mission phases and through data obtained from other spacecraft (DART, Hera, and Juventas). This choice was made early on during the project since the design phase of the algorithm had to cope beforehand with unforeseen changes.



(a) Equatorial angle.



(b) Elevation angle.

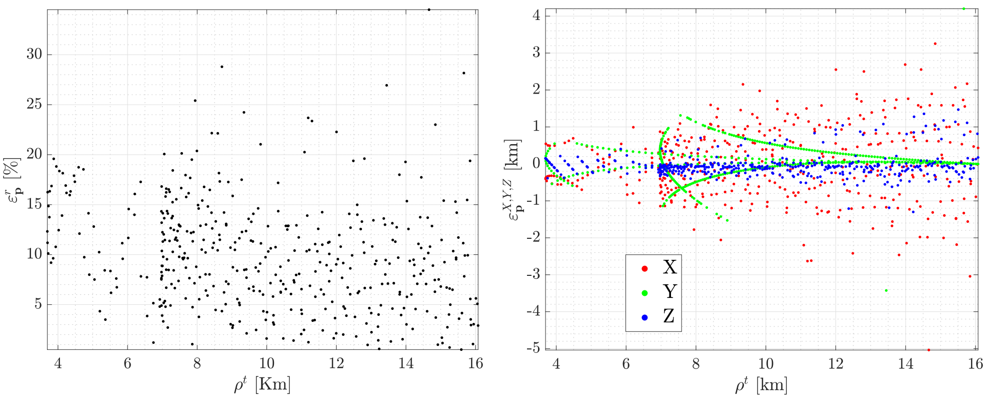


(c) Range.

Figure 6.41: Predicted (blue) and true (red) equatorial (a), elevation (b), and range (c) values as a function of time for the close-proximity trajectory.

In October 2022, the DART spacecraft arrived and successfully impacted Dimorphos. Moments before impact, it acquired crucial images about the system's appearance. This proves a vital opportunity to test the IP with updated data about the system, particularly the shape of D1.

Before the arrival of DART, D1 was expected to be a top-shaped, mild irregular



(a) $\varepsilon_{\mathbf{p}}^{r}$ error. **(b)** $\varepsilon_{\mathbf{p}}$ divided by components in $\mathcal{CAM}$.

Figure 6.42: Positioning error of the CNN represented as $\varepsilon_{\mathbf{p}}^{r}$ as function of ρ^t and divided by components in $\mathcal{CAM}$ reference frame.

body with a volume equivalent diameter of 780 ± 30 m. Its principal axes were estimated to be $832\ m \pm 3\%$ m, $838\ m \pm 3\%$, and $786\ m \pm 5\%$ and were computed using radar observations [138, 160]. After DART, a new estimate of the shape of D1 returned a more oblate body with principal axes equal to $849\ m \pm 5.6\ m$, $851\ m \pm 5.6\ m$, and $620\ m \pm 5.6\ m$ [161], the last axis suffering the biggest variation.

6.6.3.1 Challenges

A preliminary assessment using an updated version of the shape model of D1 was done after the DART impact. When new datasets were generated with the updated shapes, the entire pipeline worked as expected, apart from the WCOB method, which suffered a major drop in performance. This is caused by the data-driven function Ψ defined in Equation 6.3, used to estimate the phase angle from the eccentricity of the ellipse fitted with the same normalized second central moment to the blob of pixels associated to D1.

When considering the original scale of D1 before DART arrival from [138] ($s_z = s_0$), a second-order polynomial proved to be sufficient to represent the Ψ-e relationship, as illustrated in Figure 6.43. Such a relationship was also tested for a variation of the scale across the z-axis up to ± 5 % of the original value s_0 [119], as also discussed in Section 6.5.1 and proved to cause only minor fluctuations in the performance.

However, as it is possible to see in Figure 6.43 when considering higher values of oblateness, a clear functional relationship cannot be established anymore, considering eccentricity as the sole parameter. A specific interaction between the ellipse fit and oblate objects causes this. Increasing the oblateness of the body passing from a sphere, to D1 with $s_z = s_0$, to D1 with $s_z = 0.78 s_0$ (the value observed by DART), the eccentricity passes from having one to multiple minima. When considering a spherical object, the eccentricity of the fitted ellipse monotonically increases with the phase angle, as it is possible to see from Figure 6.45.

Because the top-shaped model of D1 prior to DART was relatively regular, this relationship could be established and exploited. However, when considering a highly oblate object, the minimum eccentricity is no longer associated with the object's projection at low phase angles. Given the irregularity of the object, a considerable offset may be present, and the minimum eccentricity is achieved with a proper combination of phase angle and point of view that does not necessarily occur at low phase angles.

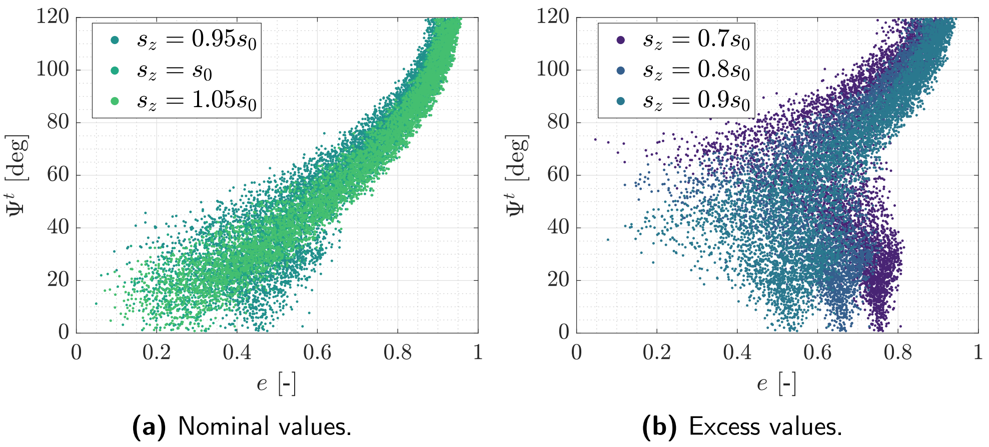


Figure 6.43: Phase angle Ψ as a function of the eccentricity of the blob of pixels associated with D1 for different values of scaling across the z-axis s_z for *5000* images randomly distributed about D1. s_0 represent the scale of D1 before the DART mission update.

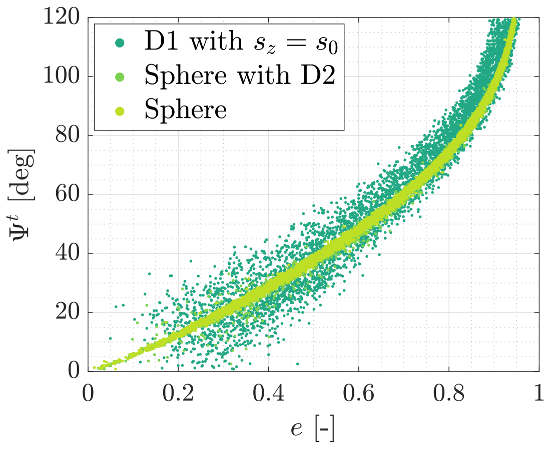


Figure 6.44: Phase angle Ψ as a function of the eccentricity of the blob of pixels associated with D1 with $s_z = s_0$ compared with a spherical body with and without D2.

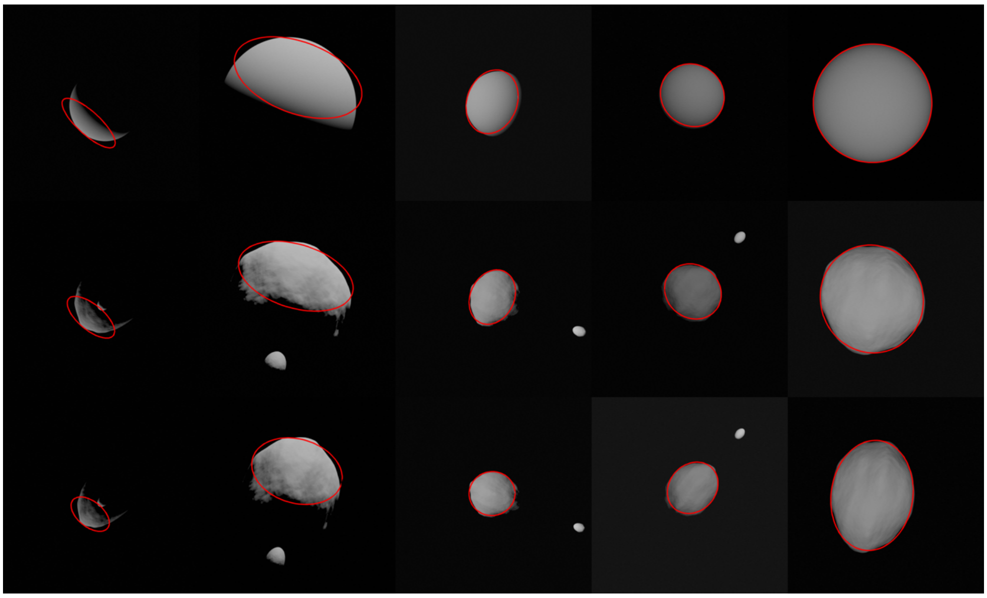


Figure 6.45: Mosaic of different views of a sphere (left), D1 with $s_z = s_0$ (center), and $s_z = 0.78 s_0$ (right) at different equatorial angles θ (from left to right 120, 90, 60, 30, and 0 deg). The red curve represents the fitting ellipse with the same normalized second-order moment.

The local irregularities over the shape of D1 and the presence of D2 in the images act purely as disturbances. This is visible from Figure 6.44, which illustrates what happens when a spherical body is substituted to D1 and subsequently when D2 is removed from the rendering software.

In conclusion, when considering more oblate shapes for D1, the eccentricity alone cannot be used as a parameter to predict the phase angle Ψ directly from images, invalidating the original design of the IP. To solve this issue, a significant update is needed on the phase angle estimation function defined in Equation 6.3 that cannot be fixed by simply adjusting the fitting coefficients. It is noted that this issue arises in the WCOB method only and that it is caused by an unforeseen change in the scale across the principal axis passing by the poles of D1, s_z, that has suffered a 22% change from new observations of the system, against the predicted 5% value considered as a requirement during the design phase.

To investigate alternative approaches, a dataset referred to as $\mathcal{DS}_{10}$ is used, consisting of images and feature vectors f_x, as illustrated in Section A.3.4. Note that images have been rendered at the resolution and range comparable to Hera's mission and assuming usage for Hera's navigation camera, but the results can be generalized for Milani's navcam. As a first step, to understand the importance of the variables that can be extracted onboard, a Principal Components Analysis (PCA) analysis is performed on the components of f_x. In Table 6.11, the elements of f_x are ordered in decreasing roles of importance according to the RRelieF metric

Table 6.11: Output of the PCA: features of f_x ordered in descending order using the RReliefF metric.

Feature	RRelieF
ν_e^{D1}	1.8487e-03
ν_{ext}^{D1}	1.5718e-03
$\log_{10}(\nu_3)$	1.3916e-03
ν_4	1.1676e-03
$\log_{10}(\nu_{area}^{edge})$	9.5524e-04
ν_{circ}^{D1}	8.6640e-04
ν_e^{edge}	6.0699e-04
$\log_{10}(\nu_{per}^{edge})$	5.4506e-04
ν_{ext}^{edge}	2.3647e-04
$\tanh(\nu_1)$	7.1865e-05
$\tanh(\nu_2)$	-2.9021e-05
ν_{circ}^{edge}	-2.3132e-04
$\log_{10}(\nu_{area}^{D1})$	-3.6348e-04
$\log_{10}(\nu_{per}^{D1})$	-1.0600e-03

[162]. Note that e is still considered the most important feature, as in the original relationship in Eq. (6.3).

6.6.3.2 Proposed approaches

Three different methods are investigated in substitution of Equation 6.3: Polynomial Chaos Expansion (PCE), features-based NNs, and image-based CNNs.

PCE, and arbitrary Polynomial Chaos (aPC), have been introduced and exploited for uncertainty quantification. However, recently PCE has been proven to be an effective technique also in other fields, e.g., it has been used to successfully propagate all-in-once a bundle of trajectories in a deterministic setting [163]. A wider use for PCE in data-driven approaches can be devised since it shows some useful properties. As a matter of fact, PCE (and aPC) can be used as an effective interpolation method, not requiring to define a-priori the interpolant functions but selecting them automatically starting from the input samples, so that they possess spectral convergence with respect to the input variables. Furthermore, the same input samples can be used to find the generalized Fourier coefficients, exploiting the least-squares approximation technique. A more detailed overview of the theory behind the PCE can be found in [164], while the focus of this section is set to its application for the problem at hand.

Practically,PCA is performed on the training dataset of feature vectors f_x associated with every image, constituting the input for PCE. The output of the PCA is then fed to aPC to build the orthogonal basis since feature distribution is not known a-priori [164].

The phase angle Ψ is then estimatd as :

1. All the features extracted from the images f_x are used as inputs for the PCA to find the principal components coefficients C, and the explained variance s;
2. Principal components are computed as $\xi = Cf_x$ and sorted by their explained variance;
3. The first dth variables are used to estimate the statistical raw moments and, in turn, the polynomial basis coefficients $a_l^{(k)}$, and the basis polynomials
4. The principal components associated to the first $M = 4000$ images are used to compute the PCE coefficients $\mathbf{c}_\alpha$

During testing, the predicted values are then obtained by using the following equation over 500 samples :

$$\hat{\mathbf{x}}(\xi) = \sum_{\alpha \in \Lambda_{p,d}} \mathbf{c}_\alpha \psi_\alpha(\xi) \tag{6.9}$$

where $\hat{\mathbf{x}}(\xi)$ is the quantity of interest, $\Lambda_{p,d}$ is a set of the multi-index of size d and order p defined on nonnegative integers, $\xi = [\xi_1, \ldots, \xi_d]$ is the set of input random variables, in which each element ξ_i is an independent identically distributed variable. The basis functions $\{\psi_\alpha(\xi)\}$ are multidimensional spectral polynomials [164].

This methodology is referred to as *PCE-Full*. An alternative one, referred to as *PCE-Res*, can be devised to perform the PCA only on the first d-th features, following the sorting order given by point 2. This approach can help the IP when performed onboard since it reduces the number of features that should be extracted from each image. Moreover, as it is possible to see in [164], considering up to nine features from f_x yields the minimum variance for both *PCE-Full* and *PCE-Res*. This value is thus considered for comparing PCE-based methods and the others.

For what concerns the NN approach, a hyperparameter search is performed varying the number of hidden layers, the number of neurons, and the activation function of a simple feed-forward NN using the Matlab regression learner application. The first and last layers of the network are made, respectively, of 14 and 1 neurons. During training, PCA is enabled on the input variables. As the output of the hyperparameter search, the best architecture is made of three hidden layers, made respectively of 124, 8, and 46 neurons, all using ReLU as an activation function. Similarly to Section 6.6.1, the NN is used mainly to investigate whether or not a network as a universal interpolation function can yield better results than a polynomial interpolator such as PCE.

Images-based CNNs architectures are also investigated in the form of CELM and CNN (see Section 2.2.2 and Section 2.2.3). The purpose is to determine whether or not a better estimate can be performed directly over the images by using implicit features, randomized, and optimized kernels in the convolutional layers. This also hints at the appropriateness of the 14 features selected to represent each image as f_x.

The same training strategy illustrated in Chapter 5 is adopted to train the CELM and CNN. *540* hierarchically organized convolutional pooling architectures are generated and trained using the CELM paradigm. The large volume of architecture is the result of a thorough architecture design search involving the weights and bias initialization strategy (Random, uniform, orthogonal), the type of activation function (none, ReLU, leaky ReLU, tanh, sigmoid), the pooling strategy (mean or max), the number of sequences of convolution, activation function, and pooling (from 2 to 6), and the values of the regularization parameter of the least-square problem (from 0.0001 to 10000 in increasing order of magnitudes).

After finding the optimal architecture setup, the architecture is trained as a traditional CNN using MBGD methods varying the value of batch size (B), learning rate, and epochs. The best-performing architecture is then selected as the one represented by the weights and biases at the minimum value of mean squared error on the validation set. The architecture of the CNN (which up to the fully connected layer FC is shared with the one of the CELM) is summarized in Table 6.12 using TensorFlow 2.10 notation.

To enable a fair comparison between all methods considered, the first *4000* samples of $\mathcal{DS}_{10}$ are always considered for training while the remaining *500* and *500* samples are considered respectively for validation and testing.

6.6.3.3 Results

In Figure 6.46, a comparison of the histogram errors of the different methods using ε_{ψ} as metric is visualized. The original performance of the WCOB with the tuning coefficient obtained with the procedure illustrated in Section 6.2.2.2 considering only ν_e^{D1} is also represented for completeness.

The performance of all methods is also summarized in Table 6.13 in a quantitative form, reporting for each method the mean, variance, Q_{67}, and Q_{95} values. Considering both Figure 6.46 and Table 6.13, the alternative approaches can be divided into three main groups. The first is represented by the WCOB and CELM, performing poorly, as expected for both cases. The second one is represented by the PCE-based methods and the NN, both achieving similar performance (also, similar w.r.t the original formulation of the WCOB with the more regular shape before DART, see Table 6.5 and Table 6.7). Finally, the third group is represented by the CNN, which considering the small variance in the ε_{ψ} error, outperforms all previous methods.

Of the different methods considered, the CNN performed the best, hinting that added filtering capabilities and the end-to-end approach played a role in generating more successful feature vectors, as previously seen in Section 6.6.1. On the other hand, PCE has demonstrated to be an excellent alternative even outside its traditional field of application, retaining similar performance with respect to the previous implementation of the IP and against NN approaches. It is also noted that the PCE implementation was limited to 3rd-order polynomials with symmetrical expansions. Differential expansions could have provided better performance.

Table 6.12: Architecture of the six layers CNN. The *max* pooling strategy and the *normalized ReLU* are used. The networks has a total count of *2,884,737* parameters (11.0 MB).

ID	Layer type	Output Shape	Parameters	Connections
I	InputLayer	($\mathcal{B}$, 128, 128, 1)	0	C1
C1	Conv2D	($\mathcal{B}$, 128, 128, 16)	160	A1
A1	nReLU	($\mathcal{B}$, 128, 128, 16)	0	P1
P1	MaxPooling2D	($\mathcal{B}$, 64, 64, 16)	0	C2
C2	Conv2D	($\mathcal{B}$, 64, 64, 32)	4640	A2
A2	nReLU	($\mathcal{B}$, 64, 64, 32)	0	P2
P2	MaxPooling2D	($\mathcal{B}$, 32, 32, 32)	0	C3
C3	Conv2D	($\mathcal{B}$, 32, 32, 64)	18496	A3
A3	nReLU	($\mathcal{B}$, 32, 32, 64)	0	P3
P3	MaxPooling2D	($\mathcal{B}$, 16, 16, 64)	0	C4
C4	Conv2D	($\mathcal{B}$, 16, 16, 128)	73856	A4
A4	nReLU	($\mathcal{B}$, 16, 16, 128)	0	P4
P4	MaxPooling2D	($\mathcal{B}$, 8, 8, 128)	0	C5
C5	Conv2D	($\mathcal{B}$, 8, 8, 256)	295168	A5
A5	nReLU	($\mathcal{B}$, 8, 8, 256)	0	P5
P5	MaxPooling2D	($\mathcal{B}$, 4, 4, 256)	0	C6
C6	Conv2D	($\mathcal{B}$, 4, 4, 512)	1180160	A6
A6	nReLU	($\mathcal{B}$, 4, 4, 512)	0	P6
P6	MaxPooling2D	($\mathcal{B}$, 2, 2, 512)	0	FC
FC	Flatten	($\mathcal{B}$, 2048)	0	D1
D1	Dense	($\mathcal{B}$, 512)	1049088	DO1
DO1	Dropout	($\mathcal{B}$, 512)	0	D2
D2	Dense	($\mathcal{B}$, 512)	262656	DO2
DO2	Dropout	($\mathcal{B}$, 512)	0	O
O	Dense	($\mathcal{B}$, 1)	513	

Table 6.13: Performance comparison between the methods considered for estimating Ψ with the updated shape.

Metric	**WCOB**	**PCE-Full**	**PCE-Res**	**NN**	**CELM**	**CNN**
μ [deg]	8.349	0.666	0.580	0.899	-1.453	0.854
σ [deg]	25.026	9.168	9.560	8.472	17.145	4.832
Q_{67} [deg]	24.963	3.447	3.863	3.169	6.310	2.690
Q_{95} [deg]	35.029	16.651	16.869	15.433	26.161	8.724

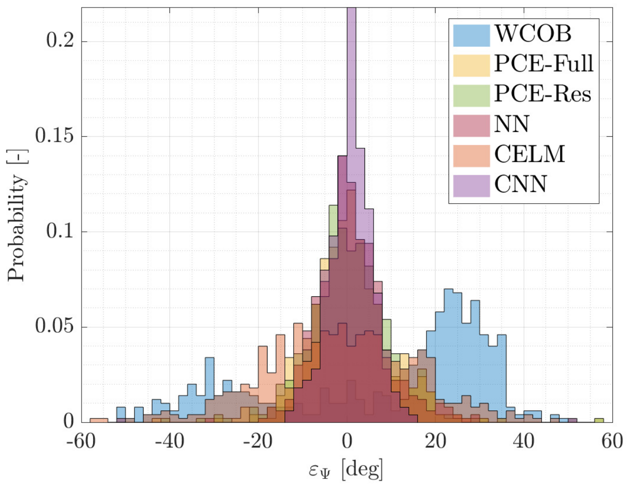


Figure 6.46: Histogram comparison between the methods considered in this section for estimating Ψ. Binwidth is *2* deg.

Ultimately, PCE will be considered for onboard implementation in place of Equation 6.3 given its simplicity, flexibility, and satisfactory level of performance compared with the other methods analyzed.

6.7 Related works

The material presented throughout this chapter is an exhaustive but concise overview of the most important works related to the Milani mission, with a particular focus on its IP and GNC subsystems. These activities have spanned without interruptions from Winter 2019 until Summer 2023, covering the entire duration of the Ph.D. activity.

For simplicity, this manuscript does not report a significant portion of the contributions to the Milani mission. Other contributions to Milani that focus on different aspects that may interest the reader are briefly discussed below.

The same process of designing close-proximity operations in a binary asteroid system that has been applied in Milani is discussed at length in [154] while the preliminary mission analysis and design of the GNC of Milani, which constituted the proposal and phase 0 design, are illustrated in [153]. The design pipeline adopted for the mission, both for the mission analysis, image processing, and GNC subsystems, is illustrated at a high level in [165].

Overviews of the mission status at different stages have also been presented in [157, 166] while the first in-depth publication of the IP and GNC has been presented in [158]. In [167, 168] and most importantly in [169], the important relationships between the trajectory design process, orbit determination, and flight operations constraints and requirements are described in detail.

In [119], the IP is tested with images-in-the-loop using the TinyV3RSE facility and a representative camera for Milani. Similar tests, but also including the GNC running on a representative processor, have been discussed in [170]. Finally, the interested reader is also directed to [155, 156] for details about the VISTA and ASPECT payloads of the Milani CubeSat.

Finally, working on a real mission proved a unique opportunity to testbench data-driven and ML algorithms on a scenario of interest with real implications. This duality between work performed for the mission and research focused on how ML algorithm could be exploited to improve performances has proven very valuable. To maximize such analyses, the Milani mission and the Didymos system, in general, have often been considered target scenarios for many of the research applications investigated during the Ph.D.

For example, in [171, 172], a Milani-like CubeSat was assumed to simulate landing and bouncing trajectories in Blender over the crater's region of Dimorphos. D1 has been considered as the target body for the works focused on semantic segmentation in [101, 104, 134, 141] while Milani and the Didymos system have been considered as test cases in [32, 173, 174] to test ML approaches for IP and navigation purposes. Finally, in [175], object recognition algorithms that use ML approaches have been investigated as alternatives to the algorithm presented in Section 6.2.1.

6.8 Final remarks

In this chapter, an in-depth overview of the Milani mission, the design of its onboard IP and GNC subsystems, the possible enhancement obtained by using ML approaches, and the challenges faced towards real operations have been extensively discussed. What follows is a list of final remarks.

- Milani is a 6U CubeSat that will visit the Didymos binary system in 2027. As part of its objectives, it will perform a technology demonstration experiment, showcasing the capability of its semi-autonomous vision-based navigation subsystem.
- The advanced functionalities of the IP pipeline of Milani are based on hand-crafted data-driven scattering functions that generate phase angle, range, and centroid estimates. These functions are explicitly defined within a complex IP pipeline, and their coefficients are tuned by using datasets of images of the system in representative conditions.
- The choice to employ mixed data-driven functionalities with traditional ones has been motivated by the irregular shape of D1 and by the expected simplicity of adapting the algorithm to its actual shape once it is imaged, first by the DART in 2022, and then by Hera in 2027.
- The reference data-driven approach has been investigated against ML ones based on NN, CNN, CELM and other data-driven ones such as PCE. Ex-

plicit features-based methods always performed substantially worse than implicit ones based on convolutional architectures. In general, it has been observed that performances are substantially boosted when extensive filtering capabilities are applied to the input images.

- Following a substantial update over the shape of D1 (22% change over the *z* axis compared to an expected worst-case scenario of 5%), a core functionality of the reference IP pipeline of Milani in estimating the phase angle suffered a drastic drop in performance. A relevant change had to be made in the pipeline to address this issue. This proved an essential lesson learned for data-driven methods and a consideration for future works. It is noted that the change was well outside the requirements and that even an entirely traditional pipeline would have suffered from extensive re-design.
- Having tested the IP pipeline under various conditions by considering different geometrical configurations, hardware settings, noise characteristics, and shape updates gives more confidence in the robustness and reliability during flight operations. These activities proved fundamental in incrementally validating the design of the IP and GNC.

7 Conclusions

"The most exciting phrase to hear in science, the one that heralds new discoveries, is not 'Eureka!' but 'That's funny...' "

Isaac Asimov

This chapter groups together the main findings of the research activity illustrated through the manuscript. The key technical findings are reported in the final remarks section at the end of each chapter. This chapter reports only the main high-level conclusions that answer the research questions introduced in Section 1.3. The chapter's concluding section also briefly comments on future works and improvements.

7.1 Conclusions

1→*Which strategies could be adopted to create the high-fidelity data-label pairs required for the supervised learning of data-driven methods?*

This question has been answered in Chapter 3. The absence of data is particularly challenging for small bodies, for which only a limited sample of visual images from previously flown missions exists. Moreover, the availability of the images, the constrained geometric and illumination conditions, and the limited size of the datasets pose stringent limitations for developing data-driven algorithms. Additionally, it is highlighted the significance of the quality of the labels, which is as important as the quality of the input images. In some cases, as for image segmentation, labels are not even readily available in the existing datasets. To counteract these issues and to be able to generate datasets in support of data-driven methods, three different strategies have been adopted: one using an artificial environment to generate synthetic renderings, one HIL setup exploiting a high-resolution screen in an optical facility, and one HIL setup using a terrain analog facility with slopes, craters, and boulders. Manual labeling of existing images has also been explored as a possibility. However, it has not proven a flexible strategy for generating a large dataset with the necessary labels, especially when considering a segmentation task.

1.a$\rightarrow$***What is the most cost-effective and flexible strategy to generate data in support of training, validation, and testing of data-driven algorithms?***

The use of an artificial environment resulted in the most flexible and cost-effective strategy to generate a large number of image-label pairs inexpensively. The design of such an environment resulted in the development of an ecosystem of tools for data generation, having CORTO at its core. CORTO is a comprehensive and versatile tool whose capabilities span rendering, noise modeling, hardware-in-the-loop testing, and post-processing, enabling researchers and engineers to simulate realistic scenarios. The tool proved critical in allowing uninterrupted image generation capabilities over four years, enabling work across research, project, and mission activities. Since the absence in the literature of reliable image similarity metrics for celestial body images, validation of the images remains an open point. Nonetheless, histogram comparison, RMSE, and SSIM metrics have been used to validate CORTO capabilities, providing satisfying results for use by onboard image processing algorithms. The other HIL setups proved too rigid for image generation for ad hoc activities and are regarded as more suitable for dataset generation instead, which is particularly useful for validation activities, expanding the capabilities of CORTO.

1.b$\rightarrow$***To what extent can existing open-source solutions be used to support dataset generation?***

Blender and Python have proven fundamental in enabling a development free of license restrictions for CORTO and the other tools illustrated in Chapter 3. In particular, Blender has proven to be an exceptional and competent tool for this specific application. However, as Blender is usually not adopted for this particular use, several limitations have been encountered. For example, material parameters or environmental settings are often expressed for usage by an artistic community and not an engineering one. This often creates a gap in the interfaces and capabilities of the tool that needs to be approached with an inventive perspective. This mentality has proven fundamental in finding ad hoc solutions and tricks that could work when simulating small body images (for example, the hair particle system used to scatter boulders over the surface randomly). Fortunately, this inventiveness is supported by many tutorials and by the extensive and vibrant community of developers using Blender. Image validation has been tackled but remains an open, existing drawback of CORTO illustrated in this work, shared among all other celestial body rendering tools.

2$\rightarrow$***To what extent can onboard applications benefit from enhanced data-driven image processing methods?***

This question has been answered in Chapter 4, Chapter 5, and Chapter 6 with the development and application of various networks for different image processing tasks.

2.a$\rightarrow$***What are the most promising image processing tasks that can be substituted or augmented?***

Image segmentation, visual-based navigation, parameter estimation, object recognition, and shape classification have been addressed as promising tasks that can be enriched by machine learning and data-driven approaches. The most exciting of these has been identified in image segmentation and visual-based navigation. Segmentation enables a complex understanding of the surrounding environment, which can be greatly exploited onboard for various tasks such as, but not limited to, autonomous scientific acquisition and tracking, hazard detection, and navigation. Navigation enables the autonomous positioning of the spacecraft around a small body, which is a critical capability to avoid collision with the same and to allow autonomous, cost-effective operations.

2.b→***What level of performance can be achieved compared to traditional approaches?***

ML approaches have been reported to consistently perform better than traditional ones, confirming an established trend in the general computer vision domain. In particular, those using hand-crafted features have been observed to perform better than traditional ones but are consistently outperformed by those using learned features. Regarding the type of architectures investigated, CNNs consistently outperforms all others. CELM-trained networks have demonstrated to be valid alternatives only in a limited number of cases, specifically when regular shapes are being considered. Nonetheless, CELMs proved pivotal in establishing a robust bootstrap training methodology widely adopted to efficiently explore the architecture design space of the networks. In general, complex training strategies have been investigated to take care of network learning, resulting in higher performance plateaus and better generalization capabilities. Concerning navigation tasks, the choice of the labeling strategy and reference frames used has proven to be fundamental in defining more straightforward, learnable mappings between images and labels. As typical for optical systems, the positioning error suffers the most in the radial direction, which can be lowered by complementing optical observations with data from range-finders sensors such as LiDARs. Interestingly, CNNs have proven flexible and robust both as local and global features-based methods. This indicates that CNNs are flexible enough to specialize with patterns of local features and the shape outline, depending on the properties of the target body. Increasing the number of images used for the position estimation (in combination with recurrent architectures) has improved the position estimate only marginally while allowing for velocity estimations only with the inclusion of LiDAR data. Moreover, noise from real images does not seem to affect the network's robustness significantly. This was attributed to the domain randomization strategies put in place and, in some cases, by the choice of the activation functions. Finally, some of these architectures made use of pieces of already existing open-access networks repurposed for the tasks at hand. On a generic note, many architectures are being increasingly made freely available for generic-purpose computer vision applications. The use of these highly performing architectures for space applications could be efficiently exploited both for semantic segmentation and relative navigation.

2.c→***What are the current drawbacks and bottlenecks for their adoptions***

in real missions?

Albeit the improvement in the performance, several limitations and drawbacks have been identified by the use of the aforementioned data-driven methods. Input images always require resizing to be efficiently used by the available hardware for training and inference. This drop in resolution from image acquisition affects the performance. The quality of the dataset generates a significant portion of the network performance. For this reason, particular care has been put not only in the network design but also in the dataset properties and quality of the inputs. In a certain sense, the pipeline of the algorithm includes the dataset, whose design shall not be overlooked too quickly. These factors have been extensively addressed in this manuscript. In some cases, relevant domain gaps have been observed between networks trained on synthetic images tested in inference on real images from previously flown missions. These have been particularly relevant for segmentation tasks that relied on manual labeling. The assessment of the domain gap is still an open point when performed directly on images and has often been assessed only indirectly as a byproduct of the network's performance with real images. The explainability of the performance has also been briefly investigated for NNs, but remains another open point for future investigations, especially for CNNs. The current approach adopted in this work has been based on the extensive testing of the network capabilities with inference tests designed to characterize network performance in scenarios never seen during training. This common-sense approach has proved fundamental in better understanding network capabilities. However, it lacks a mathematical foundation that could encourage the adoption of the methods described in this work for actual implementations. Moreover, hardware implementations have not been addressed in this manuscript, which focuses on characterizing the potential performances without considering the compatibility of the designed networks on existing space-qualified hardware for image processing. Finally, it is commented that the adoption of ML methods in real operations for interplanetary missions is still in its infancy. With this respect, two paradigms have been investigated across the manuscript, each with its pros and cons: generalization and specialization. In a generalized approach (such as in the case of a segmentation network), it is convenient to develop a robust network that can learn shared surface features from a large dataset of body shapes, geometric, and illumination conditions. In this approach, the task is learned from a variety of training episodes, which can be sampled from distributions of artificial models. This approach is powerful since it does not assume an a-priori detailed knowledge of the target body but requires extensive training that can be performed before arrival. On the other hand, in a specialized approach, a-priori knowledge of the target body's appearance is required, at least at some preliminary level. This is necessary to generate a representative artificial environment for a tailored dataset. This approach is more readily applicable to navigation tasks that are focused on a particular body's target. This modeling can be performed with the data acquired in the preliminary phases of a mission, exploiting a similar approach as the one of the Hera, Milani, and OSIRIS-REx missions.

7.2 Recommendations for future work

Suggestions for possible future works are briefly discussed. From a data generation perspective, the lack of shared testbench datasets does not help to characterize and compare the performance between different researchers, each operating with their own tailored datasets. In particular, open access to trained networks, datasets, and data-generation tools shall be encouraged to allow direct comparisons and performance assessment on shared test benches. With this respect, only three of the ten datasets illustrated in this work are currently being made publicly available. This is the first important yet limited step towards an open-access format, which shall be pursued in future works. The alternative development with Unreal Engine should also be investigated for what concern CORTO and the use of Blender. It would be interesting to explore what kind of capabilities currently implemented in CORTO could be replicated with Unreal Engine and assess what kind of improvements could this rendering software bring compared to Blender. Manual labeling has been experimented with but has proven tedious and problematic. A possible exciting opportunity would be to combine the knowledge and capability of the scientific community with the engineering one. This possibility has not been explored in this work but will be actively sought in future ones. Uncertainty quantification has been preliminary addressed both for segmentation and regression tasks (the latter are not reported in the manuscript) and is strongly believed to play a pivotal role in extending the network's capabilities. Uncertainty quantification could also play a role in fast forward the adoption of these methods in real operation scenarios, increasing the confidence in mission designer for their deployments onboard. Finally, the deployment of the architectures illustrated in this work on space-qualified processors still needs to be addressed. This represents a necessary step to demonstrate their applicability for a real mission.

A Datasets

"Data that is loved tends to survive."

Kurt Bollacker

Throughouh this manuscript, different datasets have been generated to design, validate, and test data-driven image processing algorithms. To avoid lengthening the discussion in the main corpus of the work, in this chapter, the main properties of these datasets are detailed. In Table A.1, it is possible to see a summary of all the datasets used. The datasets are divided into three groups: segmentation, navigation, and milani.

Table A.1: Summary of all the datasets used in this manuscript.

Section	Name	References	Available online
Section 4.1	$\mathcal{DS}_1$	[101]	
Section 4.2	$\mathcal{DS}_2$	[104, 176]	✓
Section 4.3	$\mathcal{DS}_3$	[141]	✓
Section 5.1	$\mathcal{DS}_4$	[177]	
Section 5.2, Section 6.6.2	$\mathcal{DS}_5$	[134]	
Section 5.3	$\mathcal{DS}_6$	[174]	
Section 6.4, Section 6.6.1	$\mathcal{DS}_7$	[32, 178]	
Section 6.4	$\mathcal{DS}_8$	[158]	
Section 6.5.1	$\mathcal{DS}_9$	[119]	
Section 6.6.3	$\mathcal{DS}_{10}$	[164]	✓

A.1 Segmentation datasets

A.1.1 $\mathcal{DS}_1$

This is the dataset used in [101]. $\mathcal{DS}_1$ is made of three datasets that can be divided into two groups: synthetic and real.

A.1.1.1 Synthetic

To generate datasets of synthetic images with an abundant and diverse presence of morphological features, an approach has been developed in [101] based on the artificial enhancement of existing shape models. The approach can be divided into two phases. First, an enhanced shape model is generated from an existing "base" model of a known small body. Second, said model and its byproducts are used to generate image-mask pairs to be used for segmentation.

Starting from the model enhancement portion, the first step consists in getting the rough shape models (i.e., the "base" models) of *9* real small-bodies from existing databases [41,42]. The models chosen are 67P/Churyumov–Gerasimenko, (101955) Bennu, (65803) Didymos, (6489) Golevka, 103P/Hartley, (8567) 1996 HW1, (10) Hygiea, (21) Lutetia, and (88) Thisbe which are referred in the rest of the section by their short names.

Their low-resolution meshes are then modified to generate higher-resolution ones. At the same time, surface roughness is simulated with texture displacement. Artificial craters are then applied on the models as different objects by using the *shrinkwrap* modifier in Blender[43]. Each crater is generated in Blender by extracting a height map from a real texture map of existing craters on Earth [44] and applying it as texture displacement on a planar mesh. Random scaling on all three axes of the crater's mesh is applied for each instance to generate multiple and diverse craters, which are then manually stitched on the shape models. The number of craters added on each model following this procedure is summarized in Table A.2.

Boulders generation follows a similar procedure. The *Rock Generator* add-on in Blender is used to generate a large, random set of rocks with varying characteristics. Each element is grouped in one of three classes depending on their qualitative size: small, medium, and large. The number of boulders is illustrated and summarized in Table A.2 for each model. The population of boulders is then applied with Blender's particle system. Different than craters, boulders are positioned automatically by the particle system, whose randomization parameters can be adjusted to obtain the desired effects. Following this procedure, the enhancement portion of the framework generates three models for each small body:

- **Clean model**: It is a model with a simplified mesh, represented by the base models without surface texture.

[41] https://sbn.psi.edu/pds/shape-models/
[42] https://3d-asteroids.space/
[43] https://www.blender.org/
[44] https://tangrams.github.io/heightmapper/

Table A.2: Summary of the craters and boulders added to each model in $\mathcal{DS}_1$.

Base Model	Craters no.	Boulders no.		
		small	*medium*	*large*
67P	2	500	-	5
Bennu	3	1000	250	10
Didymos	4	800	30	5
Golevka	2	800	-	40
103P	3	5000	30	5
HW1	2	2000	40	5
Hygiea	5	1500	100	5
Lutetia	5	1000	350	-
Thisbe	5	1000	-	20

- **Crater model**: It is a model with a refined mesh, texture, and craters. The craters are laid on the mesh but are not merged. This is fundamental in distinguishing each crater with its own individual identifier for the ray-tracing rendering engine.
- **Full model**: It is a model of the asteroid with textures, craters, and boulders. Different than the previous model, the craters are now fully merged into the mesh. Similarly to craters, boulders are not merged to obtain unique identifiers for the ray-tracing rendering engine.

These models are used in the second portion of the approach, the generation of the image-mask pairs, to produce labeled datasets. Each segmentation map is represented by a 5 layers mask. From *(0)* to *(4)* these are: *(0)* Background, *(1)* Surface, *(2)* Craters, *(3)* Boulders, and *(4)* Terminator region.

The *clean model* is used to generate the ground truth of the terminator region with a dedicated image processing pipeline in Matlab, illustrated in Figure A.1. First, a Canny edge extractor [179] is applied, providing both the sharp edge between small-body and background space and the gradual one visible in the terminator region. To avoid considering spurious edges, an acceptance mask is computed which exploits the asteroid pass index mask which does not account for shadows. However, this binary image would not exclude the body-space edges obtained via the Canny extractor. To avoid including them, the boolean acceptance image is eroded using a morphological operation with a circular structuring element [180].

The background, surface, craters, and boulders masks are easily generated using the *Cycles* ray-tracing rendering engine in Blender and exploiting different identifiers assigned to these features. This approach has been inspired by the work of [181]. Craters masks are obtained from the *crater model* while surface and boulders are obtained from the *full model*.

Once all 5 raw masks are generated, they need to be hierarchically stacked together to avoid pixels overlapping between classes. Intuitively, the hierarchy

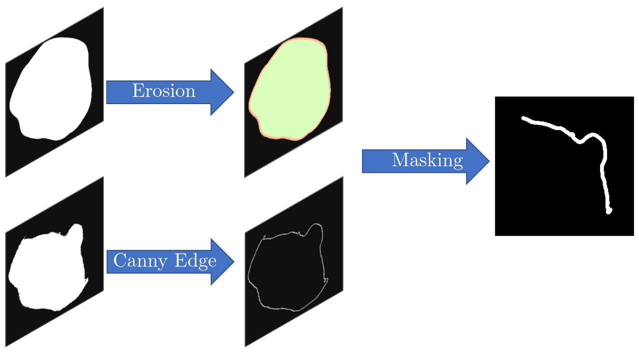


Figure A.1: Extraction of terminator region from the *clean model*.

used in order of decreasing priority is terminator, boulders, craters, surface, and background.

Finally, the grayscale image is a byproduct generated from the *full model*. The raw grayscale image obtained from rendering in Blender is further modified with the addition of artificial noise in Matlab. Gaussian noise with mean 0.1 and variance 0.0001 is added to each image, followed up by a 2D Gaussian smoothing kernel with a standard deviation of 0.1.

The image-mask pairs are then generated with random camera positions around each body sampled uniformly in a spherical shell whose radius spans $[0.4D_0, 1.3D_0]$, D_0 being the approximate range at which the body fills the FOV of the camera (assumed to be 10×10 deg). D_0 is computed as:

$$D_0 = \frac{\Gamma}{2 \tan \frac{FOV}{2}} \tag{A.1}$$

where Γ represents the maximum length of the shape model. For simplicity, an ideal pointing to the CoM of the body is assumed. This assumption is expected to have no significant impact on the performance of the network at inference time. In fact, real images acquired by the onboard sensor could go through a pre-processing centering step before being fed to the neural network.

For each acquisition, the Sun's direction is selected randomly in an angular range from the camera boresight of ± 90 deg. In such a way, a variety of illumination conditions are considered for a realistic case scenario in which a small body is seen from full to partial illumination conditions.

Following this methodology, the datasets summarized in Table A.3 are generated. $\mathcal{DS}_1^{D-1}$ is the only one used for training, validation, and testing and it is made by 7 out of the 9 available models. Two bodies, HW1 and Thisbe, are reserved for testing in $\mathcal{DS}_1^{D-2}$ and $\mathcal{DS}_1^{D-3}$. Also, $\mathcal{DS}_1^{D-3}$ represents a flyby scenario, which is of interest for the application of segmentation algorithms with unknown bodies. In particular, this scenario is characterized by a large excursion in the FOV occupancy of the target, even outside the envelope used during training. The characteristic of all synthetic datasets are summarized in Table A.3 while the ones of $\mathcal{DS}_1^{D-1}$ are detailed in Table A.4.

Table A.3: Summary of the synthetic image-mask pairs split of $\mathcal{DS}_1^{D-1}$, $\mathcal{DS}_1^{D-2}$, and $\mathcal{DS}_1^{D-3}$.

Dataset	Models	Train	Validation	Test
$\mathcal{DS}_1^{D-1}$	67P, Bennu, Didymos, Golevka, 103P, Hygiea, and Lutetia	11500	1050	1050
$\mathcal{DS}_1^{D-2a}$	HW1	-	-	1500
$\mathcal{DS}_1^{D-2b}$	Thisbe	-	-	1500
$\mathcal{DS}_1^{D-3}$	Thisbe	-	-	56

Table A.4: Summary of the synthetic image-mask pairs constituting the training, validation, and test sets of $\mathcal{DS}_1^{D-1}$.

Models	Train	Validation	Test
67P	1350	150	150
Bennu	1850	150	150
Didymos	1750	150	150
Golevka	1500	150	150
103P	1500	150	150
Hygiea	1850	150	150
Lutetia	1700	150	150
Total	**11500**	**1050**	**1050**

Finally, another dataset referred to as $\mathcal{DS}_1^{C-1}$, is used to train the encoder. $\mathcal{DS}_1^{C-1}$ is composed of the same images of $\mathcal{DS}_1^{D-1}$ but is made using different labels. Instead of having segmentation masks as labels, the body's names are considered to constitute 7 different classes. The dataset split is illustrated in Table A.5.

Table A.5: Summary of the synthetic dataset $\mathcal{DS}_1^{C-1}$.

Dataset	Models	Train	Validation	Test
$\mathcal{DS}_1^{C-1}$	67P, Bennu, Didymos, Golevka, 103P, Hygiea, and Lutetia	10880	2720	-

A.1.1.2 Real

A small dataset of real images from previously flown missions is also generated, referred to as $\mathcal{DS}_1^{D-4}$. This is comprised of 200 images randomly selected from Hayabusa I [13], Hayabusa II [16], Osiris-Rex [17], Dawn [182], NEAR Shoemaker [12] respectively of (25143) Itokawa, (162173) Ryugu, (101955) Bennu, (4) Vesta, and (433) Eros.

A selected set of 50 images from these missions is downloaded, cropped, or resized to snippets of 256 x 256 grayscale images. These have been manually labeled by the authors of [101] using the *labelbox* online tool [45]. In order to cross-validate the labeling and simplify it, some common rules have been established so that, for example, only the largest and more meaningful boulders are labeled. The labeling has been performed by two persons, each working on a 60% split of the original dataset, thus creating an overlap subset that is used as a test-bench to assess possible biases introduced by different individuals. Finally, each of the 50 manually-labeled image-mask pairs is flipped and rotated thus creating the final set of 200 pairs. The split used for training, validation, and test is illustrated in Table A.6.

Table A.6: Summary of the real image-mask pairs split of $\mathcal{DS}_1^{D-4}$.

Dataset	Models	Train	Validation	Test
$\mathcal{DS}_1^{D-4}$	Eros, Itokawa, Vesta, Bennu	140	40	20

A.1.2 $\mathcal{DS}_2$

This is the dataset used in [104] and fully characterized in [176]. $\mathcal{DS}_2$ represents an extensively annotated dataset about boulders on the surface of small bodies seen from varying illumination conditions. $\mathcal{DS}_2$ is constituted by three main datasets, referred for simplicity as: $\mathcal{DS}_2^{DS1}$, $\mathcal{DS}_2^{DS2}$, and $\mathcal{DS}_2^{DS3}$.

$\mathcal{DS}_2^{DS1}$ is composed of synthetic images of single instances of boulders positioned on a procedural-varying quasi-spherical surface. Its main purpose is to represents a single instance of a boulder positioned on the surface of a generic small body. $\mathcal{DS}_2^{DS2}$ is also composed of synthetic images, however, multiple instances of boulders are scattered across the surface of an enhanced shape model of the primary body of the (65803) Didymos asteroid. This is done to represent realistic boulder distributions scattered across a generic regular shape body. Finally, $\mathcal{DS}_2^{DS3}$ is composed of a small set of real images manually labeled from previously flown missions toward asteroids (25143) Itokawa [13], (162173) Ryugu [16], and (101955) Bennu [17]. Its purpose is to represent real boulder populations scattered across the surface of existing small bodies. $\mathcal{DS}_2^{DS3}$ coincides with $\mathcal{DS}_1^{D-4}$.

[45] https://labelbox.com/, last time accessed: 20th of October, 2023.

Table A.7: Summary of the splits of $\mathcal{DS}_2$.

Name	Acronym	$\mathcal{DS}_2^{DS1}$	$\mathcal{DS}_2^{DS2}$	$\mathcal{DS}_2^{DS3}$
Training	Tr	30181	20095	-
Validation	V	5044	5044	-
	Te_1	5044	5044	-
Test	Te_2	5000	5000	-
	Te_3	-	-	300
Total	-	45269	35183	300
Range [BU]		[2.1,10.4]	[3.9,13.0]	-
Azimuth [deg]		[-80, 80]	[-80, 80]	-
Elevation [deg]		[-30,30]	[-30,30]	-

Each of the datasets of $\mathcal{DS}_2$ is divided into training, validation, and test sets as summarized in Table A.7. The main difference between the two test sets Te_1 and Te_2 lays in the balance of the distributions of the phase angles considered. Te_1 exhibits images sampled with a balanced distribution, while Te_2 represents images sampled with randomly displace phase angles acquired from a cloud of poses around the target body. The $\mathcal{DS}_2$ dataset and all its subsets are openly available at https://zenodo.org/record/7107409.

A.1.2.1 $\mathcal{DS}_2^{DS1}$

The image-label pairs of $\mathcal{DS}_2^{DS1}$ are created using a unitary radius high-resolution spherical mesh to represent the small body, while boulder meshes are generated randomly using the *Rock generator* add-on in Blender. A set of *30* meshes, illustrated in Figure A.2, is used to represent archetype shapes of boulders, which are then singularly positioned on the surface of the body with random orientation, scaling, and albedo.

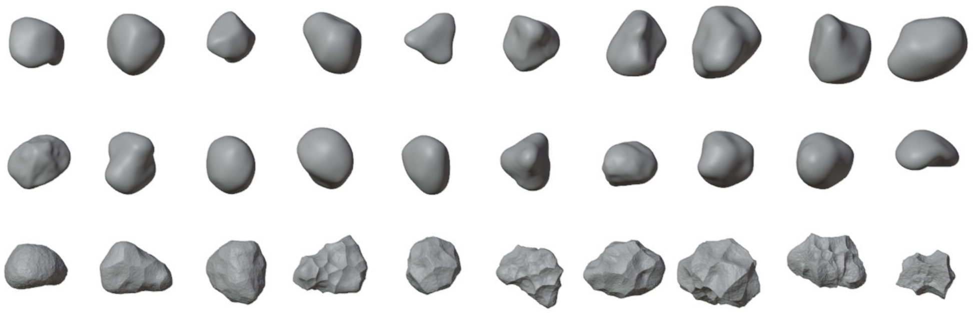


Figure A.2: The *30* archetype shapes representing single instances of boulders in $\mathcal{DS}_2^{DS1}$. From top to bottom, the *ice*, *river*, and *asteroid* classes are shown.

In order to simulate camera positions, a random cloud of points is generated around the single boulder's position, while illumination conditions are also varied randomly. During acquisition the attitude is assumed to be ideal, pointing towards the center of the boulder. Images are rendered at a resolution of 256×256 pixel, but are then post-processed in CORTO with random cropping to 128×128 size images with the addition of artificial noise. Post-processing is fundamental in making sure the boulders are not always centered in the images. Both boulders and surfaces are simulated utilizing an Akimov scattering law implemented in the shading tab of Blender. At each acquisition, the characteristics of the surface of the body are varied randomly to simulate different roughness that perturbate the environment around each boulder. With this setup, a total of 45269 image-label pairs are rendered for $\mathcal{DS}_2^{DS1}$.

Note that the masks of the boulder and surface are obtained thanks to the *Cycles* rendering engine in Blender and are generated both with and without shadows. Figure A.3 represents a sample of image-label pairs of $\mathcal{DS}_2^{DS1}$ after rendering and after post-processing.

Finally, Figure A.4 displays the camera poses during image acquisition, while Figure A.5 illustrates the main properties of the geometric quantities of $\mathcal{DS}_2^{DS1}$.

A.1.2.2 $\mathcal{DS}_2^{DS2}$

The procedure adopted to generate the image-label pairs of $\mathcal{DS}_2^{DS2}$ is in part similar to the one illustrated for $\mathcal{DS}_2^{DS1}$. The main differences are in the number of boulders positioned on the surface of the body, the size of the rendered images in (128×128), and the lack of random cropping during post-processing (only artificial noise is added to the rendered images). During rendering, instead of placing a single boulder, multiple ones are positioned on the surface of the enhanced Didymos shape model to represent a realistic boulders distribution. Once again, as in $\mathcal{DS}_2^{DS1}$, both lighting, scale, albedo, and intensity variations are randomly set to obtain a generalized dataset. A sample of image-label pairs of $\mathcal{DS}_2^{DS2}$ is represented in Figure A.6.

Finally, Figure A.7(a) displays the camera poses during image acquisition, while Figure A.7 illustrates the main properties of the geometric quantities of $\mathcal{DS}_2^{DS2}$.

A.1.2.3 $\mathcal{DS}_2^{DS3}$

Finally, the $\mathcal{DS}_2^{DS3}$ dataset is generated starting from 75, 256×256 cropped images which show clear boulders presence that has been manually labeled in [101]. Each image-mask pair is then subdivided into 4 128×128 smaller ones to reach a total of 300 samples. By design, this dataset only contains the masks of the largest boulders, as is visible in the sample in Figure A.9.

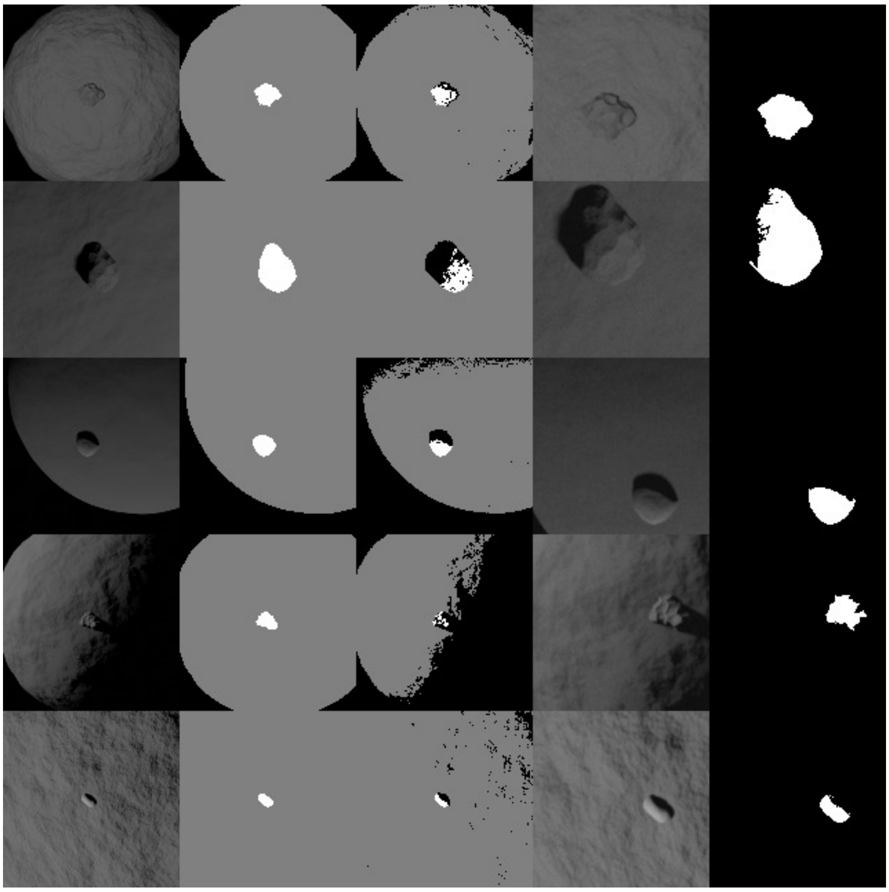


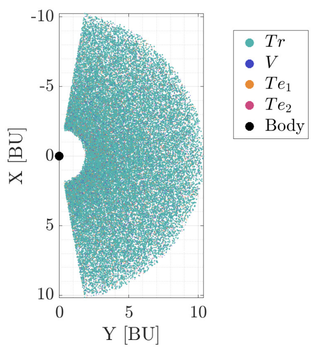
Figure A.3: Sample of image-label pairs of $\mathcal{DS}_2^{DS1}$. From left to right: 256×256 grayscale renderings in Blender, masks without shadows, masks with shadows, followed by 128×128 noisy and randomly cropped grayscale images, and relative masks with shadows.

A.1.3 $\mathcal{DS}_3$

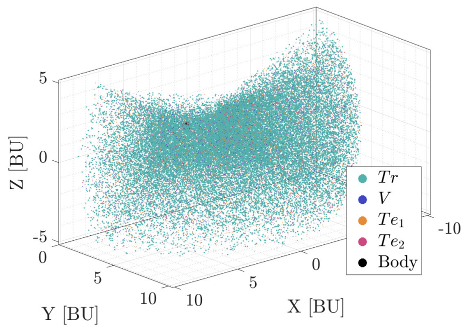
This is the dataset presented in [141]. Its geometrical properties are summarized in Table A.8 and illustrated in Figure A.10 and Figure A.11 while the dataset generation has been discussed already in Section 4.3. The $\mathcal{DS}_3$ dataset is available for download at https://zenodo.org/records/8406581.

Table A.8: Properties of $\mathcal{DS}_3$.

Parameter	$\mathcal{DS}_3$
Range [BU]	$[3.648, 14.587]$
Azimuth [deg]	$[0, 180]$
Elevation [deg]	$[-30, 30]$



(a) Top-view.



(b) 3D view.

Figure A.4: Visualization of $\mathcal{DS}_2^{DS1}$ in position phase-space in $\mathcal{W}$ reference frame. (a) top-view and (b) 3D view.

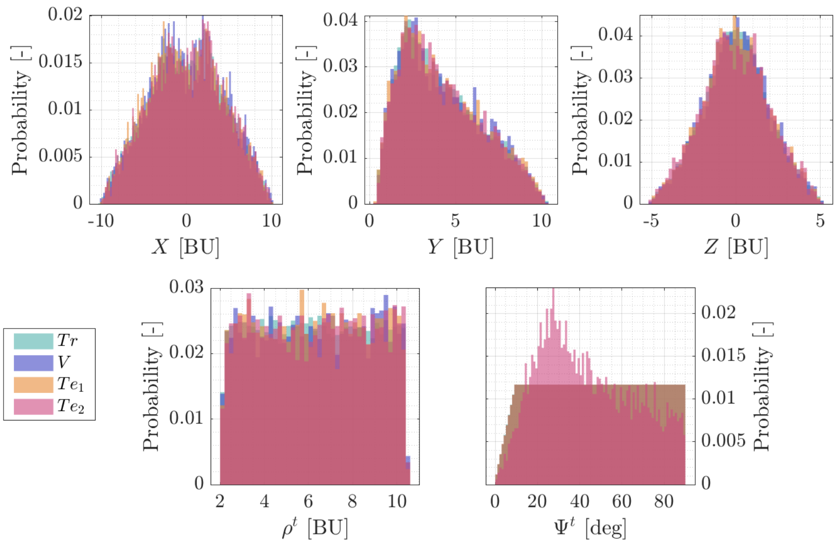


Figure A.5: Main geometric properties of $\mathcal{DS}_2^{DS1}$. The binwidth is set to *0.2* BU and *1* deg, and *probability* is used as normalization.

To generate the image-label pairs of $\mathcal{DS}_3$, a three-step procedure is adopted.

- ***Step**$_1$*: Generation of the rendering inputs. All the geometric and physical conditions that will be used to render the image-label pairs are generated for *10000* samples.
- ***Step**$_2$*: Rendering. The image-label pairs are rendered in Blender assuming a 1024×1024 pixel wide sensor with a FOV of *16* deg.
- ***Step**$_3$*: Post-processing. The image-label pairs obtained after the rendering stage are extensively post-processed, performing data pruning, data augmen-

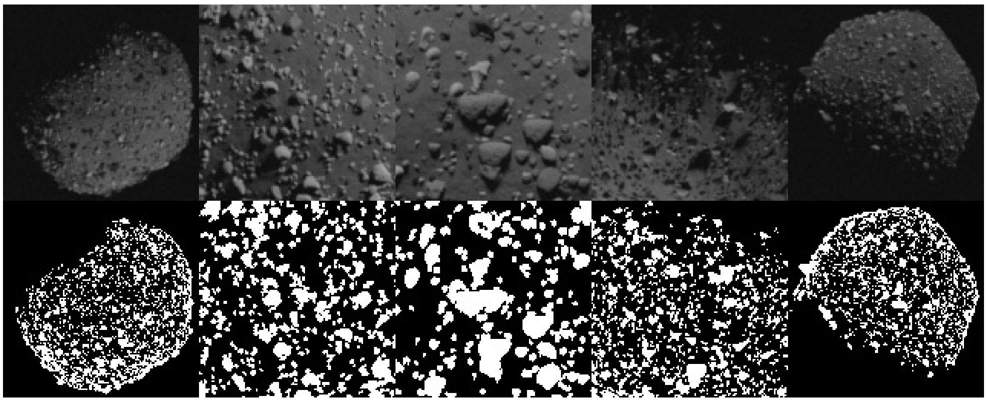


Figure A.6: Sample of image-label pairs of $\mathcal{DS}_2^{DS2}$. 128×128 noisy grayscale images (left) and relative boulder masks (right).

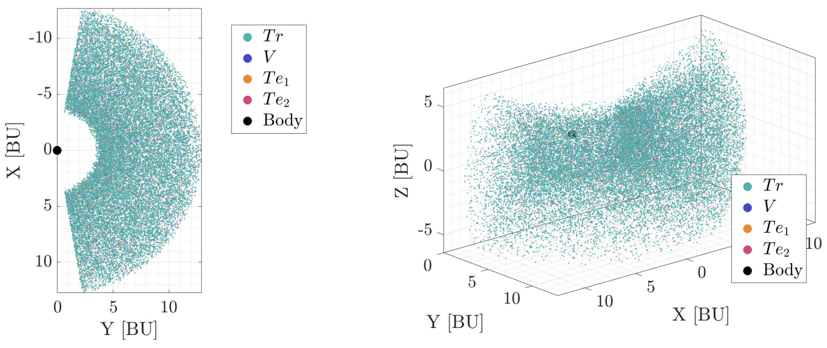


Figure A.7: Visualization of $\mathcal{DS}_2^{DS2}$ in position phase-space in $\mathcal{W}$ reference frame. (a) top-view and (b) 3D view.

tation, and data adaptation to organize the dataset for usage by the most relevant data-driven IP algorithms.

In the following sections, each of these steps is described in detail.

A.1.3.1 Rendering inputs

10000 randomly distributed camera positions X, Y, Z are generated around the asteroid in a semi-spherical shell, as illustrated in Figure A.10. The points are randomly displaced using the intervals in Table A.8.

For each position around the asteroid in the $\mathcal{W}$ reference frame, the camera pointing is offset by the quantities X_o, Y_o, Z_o, which is compounded on the ideal pointing towards the CoM of the asteroid. This is enforced to have the asteroid appear across the entire image. Additionally, two separate random rotations are used for each sample to change the orientation of the camera pointing across the boresight axis (θ_b) and the rotation of the asteroid about the Z-axis (θ_a).

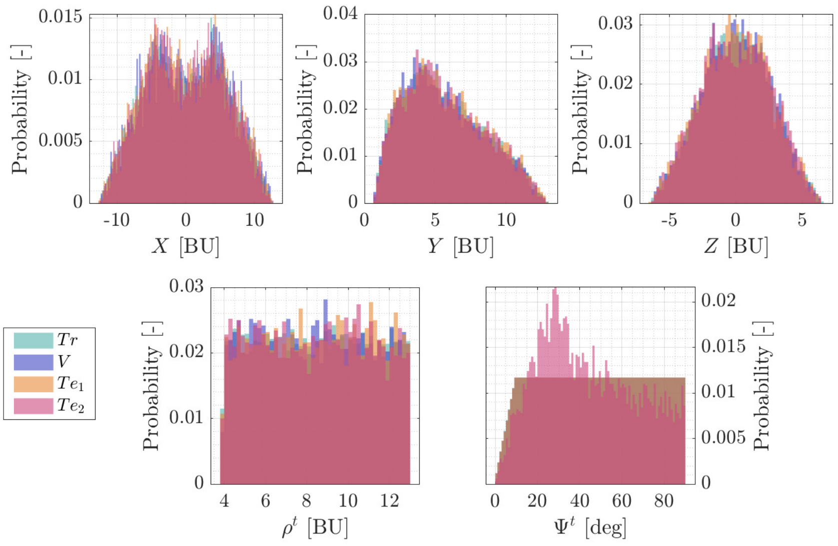


Figure A.8: Main geometric properties of $\mathcal{DS}_2^{DS2}$. The binwidth is set to *0.2* BU and *1* deg, and *probability* is used as normalization.

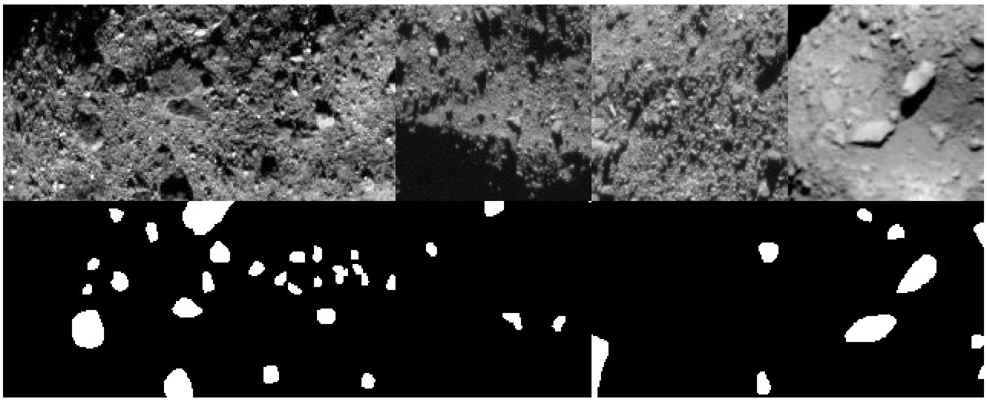
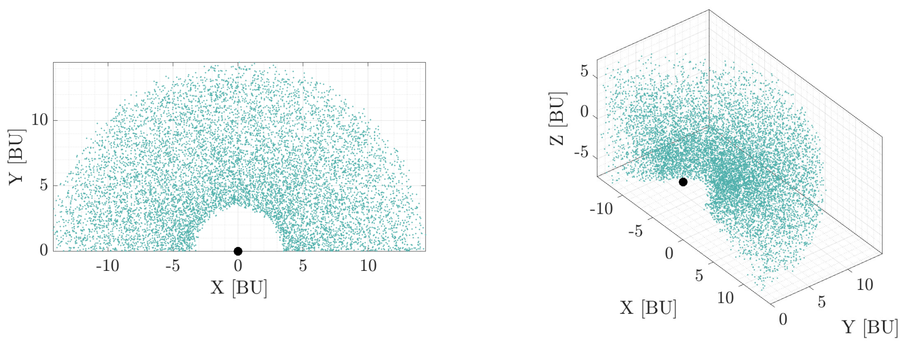


Figure A.9: Sample of image-label pairs of $\mathcal{DS}_2^{DS3}$. 128×128 real grayscale images (left) and relative boulder masks (right), manually labeled in [101].

Finally, surface properties (represented by the albedo of the surface a_s, the albedo of the boulders a_b, and the intensity of the Sun's lamp I_{Sun}) are also changed for each sample to perform domain randomization over the appearance of the body within each image under different illumination conditions. Note that in generating a_b, a relationship is established with a_s. First, a multiplicative coefficient is drawn from a standard uniform distribution between *0.7* and *3*. Such coefficient is then multiplied by a_s to obtain a_b:

$$a_b = rand(0.7, 3) \cdot a_s \tag{A.2}$$



(a) Top-view. **(b)** 3D view.

Figure A.10: Visualization of $\mathcal{DS}_3$ in position phase-space in $\mathcal{W}$ reference frame. (a) top-view and (b) 3D view.

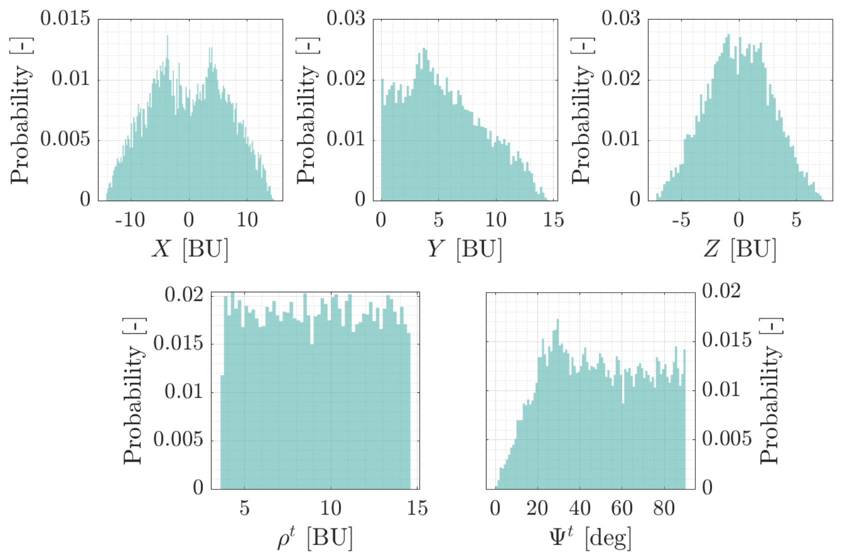


Figure A.11: Main geometric properties of $\mathcal{DS}_3$. The binwidth is set to *0.2* BU and *1* deg, and *probability* is used as normalization.

This value of a_b is the base value for the albedo of the boulders. In the next section, it will be detailed how the albedo of each boulder a'_b is changed from the input value a_b. Finally, note that the range interval from the asteroid is chosen such that once the FOV saturation distance between the camera and the body is computed (D_0), the range is set to vary between $\frac{D_0}{2}$ and $2D_0$.

A.1.3.2 Blender setup

For the $\mathcal{DS}_3$ dataset, Didymos is selected as target body. The base shape model is considered from [138], which corresponds to the shape before DART's [19] impact. This simple model is artificially enhanced with boulders of various sizes, shapes, and distributions scattered across the surface using the functionalities of the Blender's particle system.

As for the works in [101, 104, 134, 176], and similarly to $\mathcal{DS}_1$ and $\mathcal{DS}_2$, the boulders population is handled by the particle system with the settings presented in Table A.9, divided by the *Ice*, *River*, and *Asteroid* classes of the *rock generator* add-on.

Table A.9: Settings of the particle system in Blender to generate the boulder's population.

Size & Class	Ice	River	Asteroid
Small	2500	2500	2500
Medium	200	200	200
Large	5	5	5

The first two sizes in Table A.9 (Small and Medium) are indicated as "minor boulders" while the latter size (Large) is indicated as "prominent boulders". Differently from $\mathcal{DS}_1$ and $\mathcal{DS}_2$, $\mathcal{DS}_3$ specializes in the properties of the large boulders as prominent surface features that can also be exploited as landmarks for vision-based navigation.

For this reason, while the 8100 minor boulders are all rendered with the same pass index, the 15 prominent boulders have unique layers assigned to them. The IDs of the layers obtained during rendering are summarized in Table A.10.

Table A.10: Description of the layers in the segmentation mask of $\mathcal{DS}_3$.

Layer ID	Description
0	Background
1	Surface
2	Minor boulders
3-17	Prominent boulders

Once the particle system is used to generate the boulders population over the surface of Didymos, the model is considered complete and remains unchanged throughout the entire rendering procedure. This ensures that the same geometrical distribution of the surface features is seen across the dataset from different images and points of view.

While the geometrical distribution is not changed, the appearance of the surface and the boulders is varied to achieve domain randomization on the input images.

To achieve such an effect, scattering laws are applied to the surface and each boulder, whose albedo is adjusted singularly. To realize such a level of detail, the albedo of each boulder is set by modifying a_b from the input *txt* as follows:

$$a'_b = max \begin{cases} a_b + rand(-0.5, 0.5)\sigma_b \\ 0.03 \end{cases} \tag{A.3}$$

where σ_b is an arbitrary parameter set to 0.35. Note that the random value used to modify each boulder is not saved during rendering. Finally, I_{Sun} sets the intensity of the Sun's lamp for each sample. All these parameters, together with the artificial noise added in $Step_3$, ensure domain randomization over the appearance of the boulders, which should translate into robustness and generality in the data-driven IP methods that will be using the $\mathcal{DS}_3$ dataset.

A.1.3.3 Post-processing

After rendering the 10000 samples, extensive post-processing is performed. First, artificial noise is introduced to the images. The purpose of this step is not to emulate a specific camera but rather to generate noise with varying characteristics to develop a noise-agnostic IP method. The noise pipeline is developed in Matlab and is composed of the following steps applied in this order to each image:

- A Gaussian filter is used over the image to simulate a generic blur. The kernel is generated with $\sigma = 0.5$.
- Motion blur is simulated over the image by setting a variable length in pixels and an orientation equal to 0 deg.
- Gaussian noise is added with a variable mean and a variance of 0.0001.
- A variable γ correction is applied to the image.

Other effects, such as dead buckets, dead pixels, blooming, or radiation effects, are not simulated. In the pipeline, the noise added to the rendered images is represented by a static and a dynamic component. The static has parameters that stay the same for the entire dataset, while the dynamic changes for each sample. The dynamic components of the noise correspond to the pixel length of the motion blur (sampled with a uniform distribution between 0.1 and 2), the mean value of the Gaussian noise (sampled with a uniform distribution over the exponents between 10^{-4} and 10^{-1}, and the gamma correction coefficient (sampled with a uniform distribution between 0.9 and 1.1). After adding noise to the samples, each image-label pair is split into five sub-samples, as illustrated in Figure A.12 for image $00000j$.

This strategy is also referred to as the "5-way split". The first 4 subsamples are cropped from the original image, while the fifth one is obtained through a resize of the original image. The purpose of the 5-way split is twofold. First, to perform data augmentation of the samples. Second, to obtain different regions imaged at different scales and resolutions. After the split, a statistical characterization is performed to remove spurious cases. These include 2498 cases in which the image content is below a predefined threshold. A final pruning is also performed over the

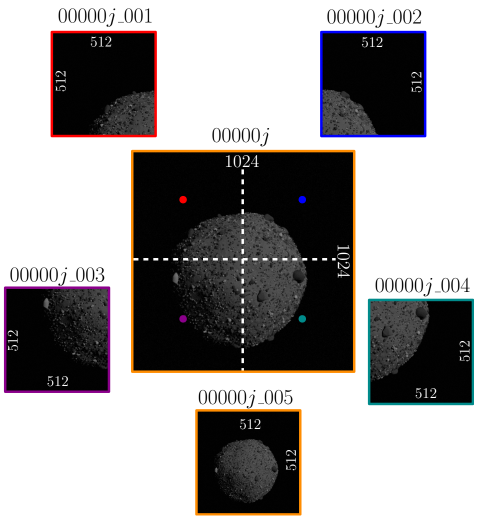


Figure A.12: Example of the 5-way split performed over an original image *00000j*.

masks of the prominent boulders with an area below a predefined threshold. This turned necessary from previous findings in $\mathcal{DS}_2$. At the end of this pruning, the split transforms the *10000* original dataset of 1024×1024 image-label pairs in one made of *47502* samples at a resolution of 512×512, which is closer to the typical size used for training convolutional networks.

Finally, a data preparation step is included to extend the applicability of the dataset. Indeed, up to this stage, only 2D masks have been handled as labels. However, some techniques may require different standards. In particular, the dataset format for the prominent boulders is made functional for You Only Look Once (YOLO) [183] architectures. To accomplish this, the masks of the prominent boulders are converted as lists of connected vertices and saved as *.txt* files.

A.2 Navigation datasets

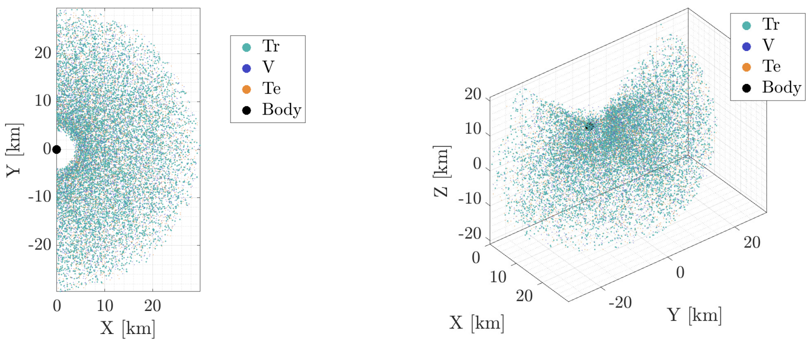
A.2.1 $\mathcal{DS}_4$

This is the dataset used in [177]. The dataset is constituted by image-label pairs. Its geometrical properties are summarized in Table A.11. The camera poses are illustrated in Figure A.13 while dataset statistics are illustrated in Figure A.14.

In $\mathcal{DS}_4$, four small bodies (Didymos $\mathcal{D}$, Hartley $\mathcal{H}$, Lutetia $\mathcal{L}$ and 67P $\mathcal{P}$) are considered, sharing the same camera poses. Adopting the same methodology used in $\mathcal{DS}_1$, artificial morphological features such as boulders and craters are added to the reference shape model of these bodies. These are considered only as sources of disturbance. Since each dataset is composed of the same set of points across different bodies, scaling is applied to each body to make sure that different bodies are all filling the FOV around *5* km.

Table A.11: Properties of $\mathcal{DS}_4$.

Parameter	$\mathcal{DS}_4^{Tr}$	$\mathcal{DS}_4^{V}$	$\mathcal{DS}_4^{Te}$
Number of images	7500	5000	5000
Range [km]	[5, 30]	[5, 30]	[5, 30]
Azimuth [deg]	[-90, 90]	[-90, 90]	[-90, 90]
Elevation [deg]	[-45, 45]	[-45, 45]	[-45, 45]



(a) Top-view. (b) 3D view.

Figure A.13: Visualization of $\mathcal{DS}_4$ in position phase-space in the $\mathcal{W}$ reference frame. (a) top-view and (b) 3D view.

To generate the input grayscale images, a camera with a 10×10 deg FOV and a sensor of 1024×1024 pixels is considered. All images are rendered assuming the ideal pointing towards the CoM of each body. The camera position associated with each image is encoded as a label using five different strategies. The first one consists of saving the optical observables linked to geometrical quantities that can be directly extracted from images. These are:

$$\delta = \textbf{CoF} - \textbf{CoB} \quad , \quad \rho \tag{A.4}$$

where δ is the difference in pixels between the CoB and CoF of the body projected in the image plane and ρ is the range from the CoM. These quantities can be used to generate a position estimate in the camera frame, which can be transformed in $\mathcal{W}$ or $\mathcal{AS}$ by simulating onboard attitude determination from a star-tracker alongside the assumption of knowledge of the rigid rotation between the inertial fixed reference frames used by star-tracker and the $\mathcal{W}$ or $\mathcal{AS}$ frames. The procedure to do so is fully explained in Section 3.1.1.5.

The second and third label encodings are represented by the spacecraft position respectively in spherical and cartesian coordinates in the $\mathcal{W}$ frame. Similarly, the fourth and fifth encodings are represented by spherical and cartesian coordinates in the $\mathcal{AS}$ frame.

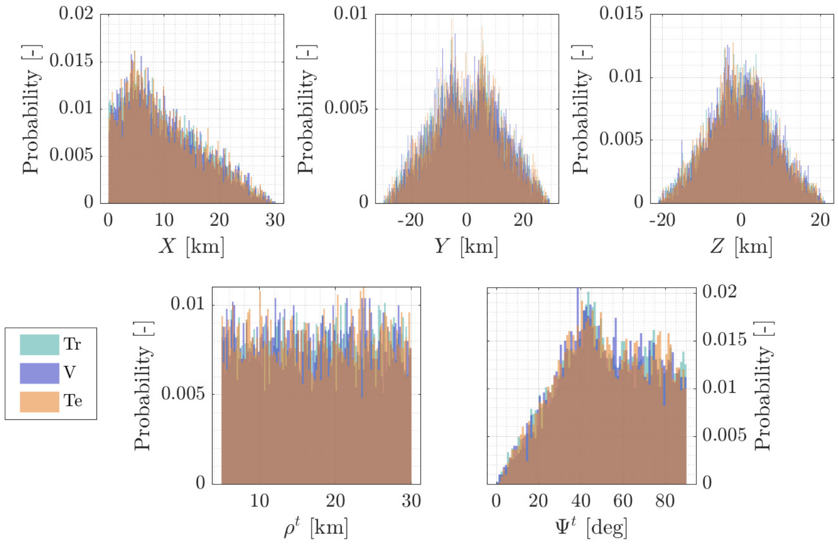


Figure A.14: Main geometric properties of $\mathcal{DS}_4$. The binwidth is set to *0.2* km and *1* deg, and *probability* is used as normalization.

By combining the four different small bodies with these five different labeling strategies, a total of 20 image-label subsets are explored in $\mathcal{DS}_4$. To navigate easily among these datasets, the shared notation represented in Table A.12 is adopted. A sample of the dataset is visible in Figure 5.12.

Table A.12: Notation used for the $\mathcal{DS}_4$ dataset.

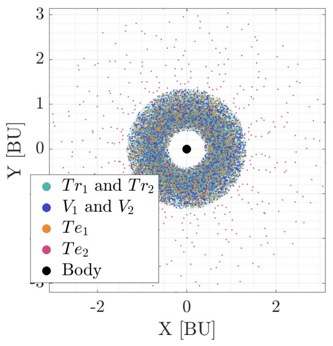
ID	Body	Frame	Labels	Notation
1	$\mathcal{D}$	-	δ,ρ	D1
2	$\mathcal{H}$	-	$\delta,\ \rho$	H1
3	$\mathcal{L}$	-	$\delta,\ \rho$	L1
4	$\mathcal{P}$	-	$\delta,\ \rho$	P1
5	$\mathcal{D}$	$\mathcal{AS}$	ϕ_1,ϕ_2,ρ	D2
6	$\mathcal{H}$	$\mathcal{AS}$	ϕ_1,ϕ_2,ρ	H2
7	$\mathcal{L}$	$\mathcal{AS}$	ϕ_1,ϕ_2,ρ	L2
8	$\mathcal{P}$	$\mathcal{AS}$	ϕ_1,ϕ_2,ρ	P2
9	$\mathcal{D}$	$\mathcal{AS}$	X,Y,Z	D3
10	$\mathcal{H}$	$\mathcal{AS}$	X,Y,Z	H3
11	$\mathcal{L}$	$\mathcal{AS}$	X,Y,Z	L3
12	$\mathcal{P}$	$\mathcal{AS}$	X,Y,Z	P3
13	$\mathcal{D}$	$\mathcal{W}$	ϕ_1,ϕ_2,ρ	D4
14	$\mathcal{H}$	$\mathcal{W}$	ϕ_1,ϕ_2,ρ	H4
15	$\mathcal{L}$	$\mathcal{W}$	ϕ_1,ϕ_2,ρ	L4
16	$\mathcal{P}$	$\mathcal{W}$	ϕ_1,ϕ_2,ρ	P4
17	$\mathcal{D}$	$\mathcal{W}$	X,Y,Z	D5
18	$\mathcal{H}$	$\mathcal{W}$	X,Y,Z	H5
19	$\mathcal{L}$	$\mathcal{W}$	X,Y,Z	L5
20	$\mathcal{P}$	$\mathcal{W}$	X,Y,Z	P5

A.2.2 $\mathcal{DS}_5$

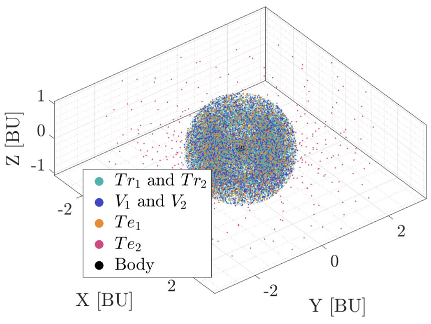
This is the dataset used in [173]. The dataset is constituted by image-label pairs, where images are represented by segmentation masks. Its geometrical properties are summarized in Table A.14. The camera poses are illustrated in Figure A.15 while dataset statistics are illustrated in Figure A.16.

Didymos ($\mathcal{D}$) and Hartley ($\mathcal{H}$) are considered in $\mathcal{DS}_5$ as representative of regular and irregular small bodies. For simplicity, Dimorphos, the secondary body of the Didymos binary system, is not considered in the renderings. A camera sensor with a FOV of 10° is assumed, and the space around each shape model is normalized based on the distance at which the maxim bounding box of the model touches the edges of the image. This normalization distance is referred to as D_0. Considering the shape models of the two bodies and the camera's characteristics, $D_0 = 4.81$ km for Didymos and $D_0 = 14.41$ km for Hartley.

Spherical shells around each body in the $\mathcal{AS}$ reference frame are obtained enforcing the geometric constraints in Table A.14. From these shells, a total of 1176 regions are carved in polar coordinates. Intervals of 0.1 BU, 15 deg, and 15 deg are considered respectively for the range, equatorial and elevation angles $(R-\theta-\phi)$, generating respectively 7 macro-classes for range and elevation angles



(a) Top-view.



(b) 3D view.

Figure A.15: Visualization of $\mathcal{DS}_5$ in position phase-space in the $\mathcal{AS}$ reference frame. (a) top-view and (b) 3D view.

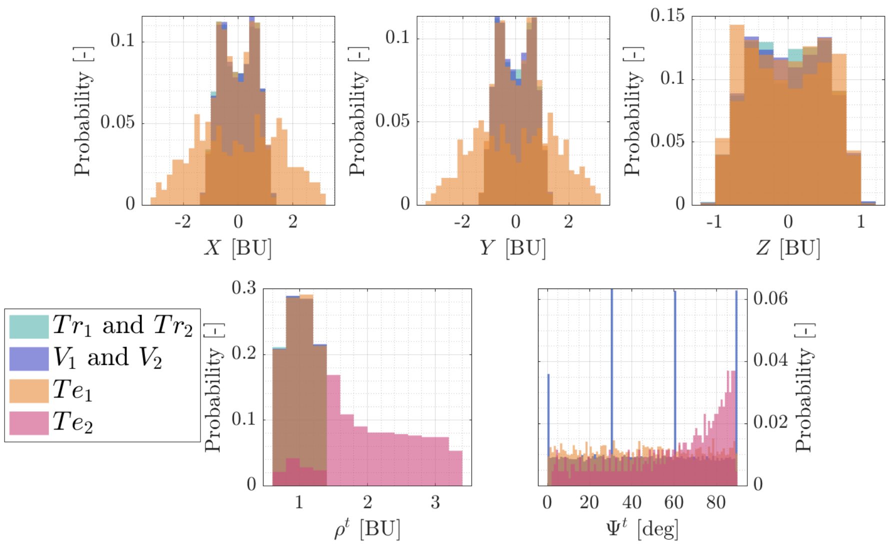


Figure A.16: Main geometric properties of $\mathcal{DS}_5$. The binwidth is set to *0.2* BU and *1* deg, and *probability* is used as normalization.

each and *24* macro-classes for the equatorial angle, for a total of $7 \times 7 \times 24 = 1176$ classes. Some of these classes are illustrated in Figure 5.14. For each class, varying illumination conditions are reproduced, assuming the Sun's direction to lie within the equatorial plane and with admissible phase angles between $\pm 90°$. The pointing is considered ideal. The splits between training, validation, and test sets are summarized in Table A.13.

Tr_1 comprises *7* image-mask pairs per class (taken at the center of the class in polar coordinates) with *7* equally spaced phase angles. V_1 and Te_1 are taken

Table A.13: Number of images per test case for each small-body shape.

Set	Number of images	SC position	Sun orientation
Tr_1	8232	Centered (Class)	Regular
Tr_2	35280	Uniform (Class)	Uniform
V_1	1500	Uniform (All)	Uniform
V_2	4704	Uniform (Class)	Uniform
Te_1	5000	Uniform (All)	Uniform
Te_2	432	Trajectory	Trajectory

with randomly selected positions satisfying the spherical shell constraints and with random illumination conditions. Tr_2 and V_2 sets are obtained by selecting 30 and 4 random positions within each class with random illumination conditions. Finally, Te_2 is made by regularly sampled positions taken from a 3-day orbit representative of an operational scenario around Didymos from the Milani mission.

Table A.14: Properties of $\mathcal{DS}_5$ for $\mathcal{D}$ and $\mathcal{H}$. $\mathcal{H}$ does not have samples for $\mathcal{DS}_5^{D-4}$, which is only considered a case for Milani's proximity operations around Didymos.

Parameter	$\mathcal{DS}_5^{D-1}$, $\mathcal{DS}_5^{D-3}$	$\mathcal{DS}_5^{D-2}$	$\mathcal{DS}_5^{D-4}$	$\mathcal{DS}_5^{D-5}$
Number of images	5000, 5000	15000, 15000	432, n.a.	1000, 1000
Range [BU]	[0.65, 1.35]	[0.65, 1.35]	[0.77, 3.34]	[0.65, 1.35]
Azimuth [deg]	[-180, 180]	[-180, 180]	[-180,180]	[-180, 180]
Elevation [deg]	[-45, 45]	[-45, 45]	[-15, 15]	[-45, 45]

The image-mask pairs are generated synthetically and processed using CORTO. Both are 128×128 pixel wide, and the images are in grayscale. At the same time, the segmentation masks assume five different discrete levels corresponding respectively to the background, body surface, craters, boulders, and terminator region. The procedure to generate the masks is the same as the one illustrated in Section A.1.1 for $\mathcal{DS}_1$.

A.2.3 $\mathcal{DS}_6$

This is the dataset used in [174]. It is divided into two portions, one used by the CELM architecture and one used by the RNN one.

A.2.3.1 CELM dataset

Following the same procedure illustrated to generate $\mathcal{DS}_1$ and $\mathcal{DS}_5$, a dataset of 12500 segmentation maps around the primary of the Didymos binary system is generated using CORTO. Each segmentation map is composed of 1024×1024 pixels, each representing a specific morphological feature with a value from

0 to 3: Background (0), surface (1), craters (2), and boulders (3). The main properties of the dataset are illustrated in Table A.15.

Table A.15: Properties of $\mathcal{DS}_6$.

Parameter	$\mathcal{DS}_6^{Tr(C)}$	$\mathcal{DS}_6^{V(C)}$
Number of images	7500	5000
Range [km]	[5, 30]	[5, 30]
Azimuth [deg]	[-85, 85]	[-85, 85]
Elevation [deg]	[-50, 50]	[-50, 50]

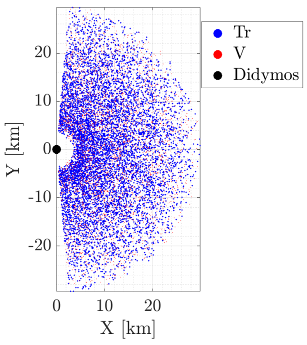
The range interval is always chosen to have the body resolved by the FOV of the simulated sensor, which is 10×10 deg wide. The segmentation maps are rendered assuming ideal pointing. They are accompanied by a set of labels that can be extracted from the image and can be used to establish the position of a spacecraft w.r.t the asteroid. Using the best-performing strategy illustrated in [177], the (δ, ρ) labels are used. Each image is thus associated with a three components vector: The first two components represent the estimated correction in pixel in the image plane between the CoB and the projected CoM (also referred to as δ or scattering correction), while the third component is the range ρ from the CoM of Didymos.

Using the same preprocessing pipeline described in detail in Section 3.1.1.5 (without the inclusion of noise in the segmentation maps), data augmentation is performed on the input, and each map-label pair is transformed into a 128×128 matrix with the asteroid not necessarily appearing centered in the frame but rather randomly displaced in it. Input and labels are also normalized in preparation for training.

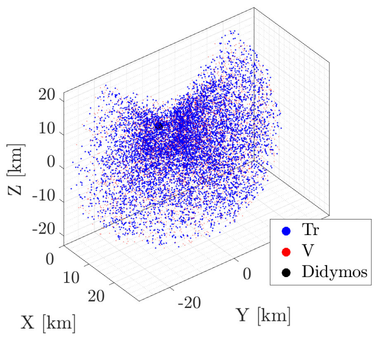
The characteristic of the points in $\mathcal{DS}_6$ is chosen as a reasonable assumption for a realistic proximity scenario and is based on previous experience gained on the design of the proximity operations of the Milani mission. The camera poses of $\mathcal{DS}_6$ are illustrated in Figure A.17 while dataset statistics are illustrated in Figure A.18.

A.2.3.2 RNN dataset

Using the dynamical model illustrated in [154] and the propagator tool developed to design the CPO of Milani, a dataset of position-velocity pairs is generated for $\mathcal{DS}_6$. The dynamical model considered three main accelerations: the gravity of Didymos, the third-body effect of the Sun, and the solar radiation pressure. Differently than the model in [154], the gravitational acceleration caused by Dimorphos, the secondary body of the Didymos binary system, is not modeled. The gravity of Didymos is modeled as a point mass for regions above 1.1 km and using a polyhedra model below this range. This threshold is clearly illustrated in the perturbation analysis in [154]. Also, instead of having the position of Didymos and the Sun



(a) Top-view.



(b) 3D view.

Figure A.17: Visualization of $\mathcal{DS}_6$ in position phase-space in the $\mathcal{W}$ reference frame. (a) top-view and (b) 3D view.

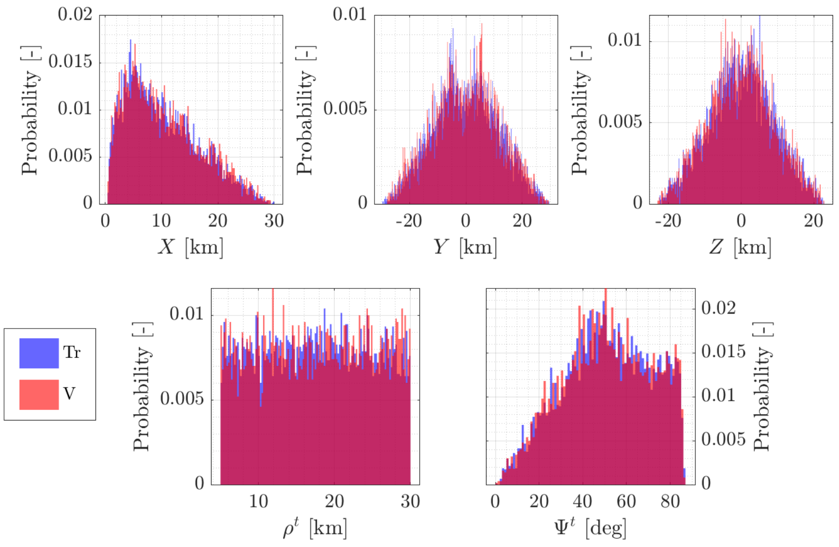


Figure A.18: Main geometric properties of $\mathcal{DS}_6$. The binwidth is set to *0.2* km and *1* deg, and *probability* is used as normalization.

resolved precisely using ESA's Hera mission kernels as in [154] at any given time, both Didymos and the Sun positions are assumed at a fixed epoch to simplify the analysis, and allow an adequate comparison between the different CPOs.

In such an environment, trajectories are designed using a strategy consisting of ballistic arcs patched together at maneuver points [169], called waypoints. Each ballistic arc between two consecutive waypoints is based on a step-wise differential correction procedure and is the result of an iterative targeting problem. First, initial conditions are set that determine the ballistic trajectory to be flown by the spacecraft. These include the position of the two waypoints, the initial epoch, and the time of flight between waypoints. Second, a restricted two-body problem Lambert's solver is used to find a suitable first guess solution for the initial velocity. Third, the initial state is propagated forward using the dynamical model. The deviation between the actual endpoint and the desired one is used to compute a correction term on the initial velocity generated at the previous step. The last step is iterated until the discrepancy between the two end states is below an arbitrarily defined threshold. More details about the differential correction scheme can be seen in [154, 169].

Using such a scheme in combination with the dynamical model, a total of 144411 trajectory points scattered across Didymos are generated. These are divided into two main groups, made of 96000 and 48411 points.

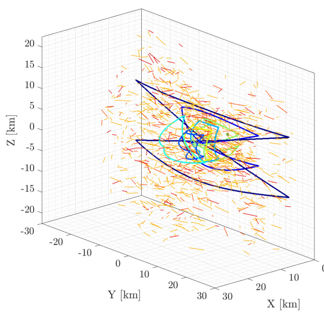
The first comprises short pieces of open trajectories computed from 800 Initial Condition (IC)s randomly distributed across the position and velocity phase space, as illustrated in Figure A.19 and Figure A.21. These trajectories are obtained from a forward propagation with a fixed timestep of 150 s for a total of 120 steps. Note that the training, validation, and test splits in Table A.16 are divided consistently in different random trajectories so that respectively 640, 80, and 80 ICs are used for each split. The main properties of the dataset are illustrated in Table A.16, while the histogram distributions are illustrated in Figure A.20 and Figure A.22

Table A.16: Properties of $\mathcal{DS}_6$.

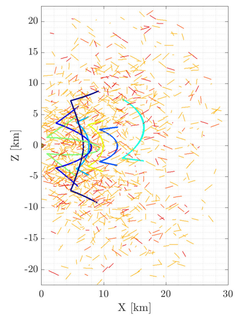
Parameter	$\mathcal{DS}_6^{Tr(R)}$	$\mathcal{DS}_6^{V(R)}$	$\mathcal{DS}_6^{Te(R)}$
Number of images	76800	9600	9600
Range [km]	[5, 30]	[5, 30]	[5, 30]
Azimuth [deg]	[-85, 85]	[-85, 85]	[-85, 85]
Elevation [deg]	[-50, 50]	[-50, 50]	[-50, 50]

The second group, made of 48411 points, comprises closed trajectories from 7 different ICs, which include intermediate maneuvers. Closed trajectories are obtained by simply constraining the first and last waypoints to be the same, using the differential correction scheme used in [154] and described before. This dataset is made of 48411 points, which are entirely used for testing. The CPOs described in this dataset are representative of possible geometries to be used in the proximity of a small body and, in particular, are representative of real CPOs, which can be adopted around the Didymos asteroid [154]. The geometries are arbitrarily chosen from experience gained on the Milani mission as well as loosely inspired from geometries seen in previous missions around small bodies such as in [184].

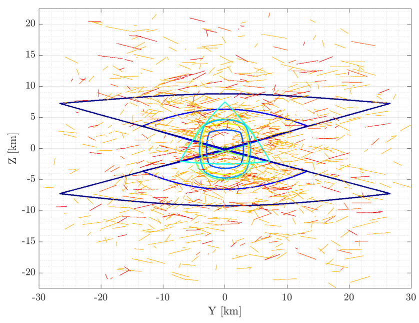
Once the position-velocity pairs have been determined from the dynamical model for all datasets, segmentation maps are generated for each of the 144411 trajectory points of $\mathcal{DS}_6^R$. Each map is then used in inference for the best CELM obtained in the previous section to generate 144411×2 position estimates (with



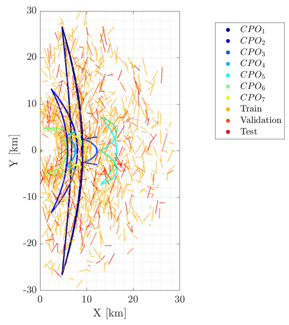
(a) 3D view.



(b) XZ view.



(c) YZ view.



(d) XY view.

Figure A.19: Visualization of $\mathcal{DS}_6$ in position phase-space in the $\mathcal{W}$ reference frame.

the use of ρ_o or ρ_l, see Section 5.3).

After this passage, the entire dataset is transformed into a sequence of fixed-interval position-velocity pairs expressed in $\mathcal{W}$ reference frame. It is noted that computing the position estimate apriori greatly simplifies and speeds up the training of the RNN while retaining its capability to be developed for an onboard application.

Each of the 807 different trajectories considered for the RNN is divided into multiple shards defined by three parameters: the time interval between position estimates ΔT (which is a multiple of 150 s), the total number of position estimates N from the CELM to be used to generate an estimate with the RNN, and the initial sample of the shard from the original trajectory j. In this paper, 5 possible combinations of $(\Delta T, N)$ are investigated, as illustrated in Table A.17, while j is rolled forward until there are enough points in the trajectory to generate a shard of N samples. For simplicity, the test sets are divided into Te_1 and Te_2 , respectively representing the random shards within the training envelope generated by the 80 ICs, and the ones of the 7 CPOs illustrated in Figure A.19 and Figure A.21.

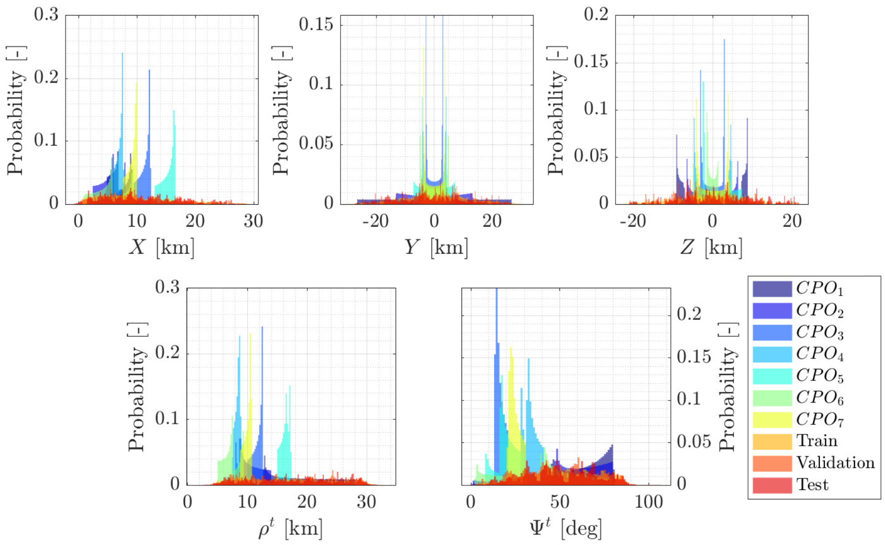
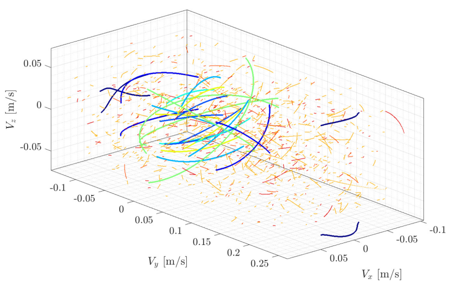


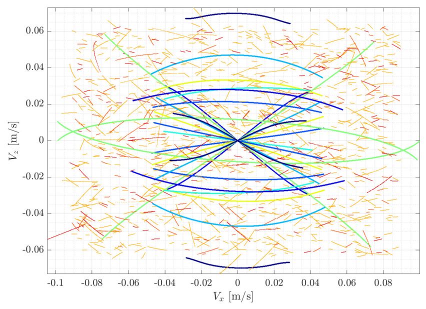
Figure A.20: Main properties of $\mathcal{DS}_6$ in the position-phase space. The binwidth is set to *0.2* km and *1* deg, and *probability* is used as normalization.

Table A.17: Number of shards in each train, validation, and test sets for each possible combination investigated.

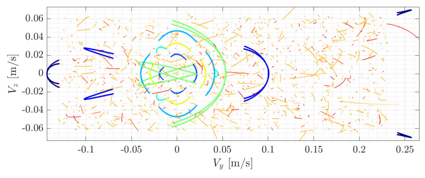
ΔT	N	Train	Validation	Te_1	Te_2
150	5	74240	9280	9280	48303
150	30	58240	7280	7280	47628
1800	5	46069	5760	5760	47104
3600	5	15337	1920	1920	45796
5400	3	30685	3840	3840	46432



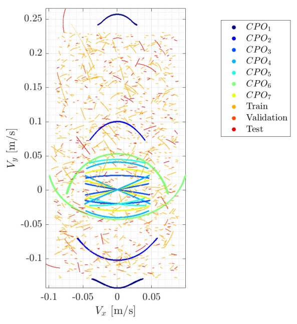
(a) 3D view.



(b) XZ view.



(c) YZ view.



(d) XY view.

Figure A.21: Visualization of $\mathcal{DS}_6$ in velocity phase-space in the $\mathcal{W}$ reference frame.

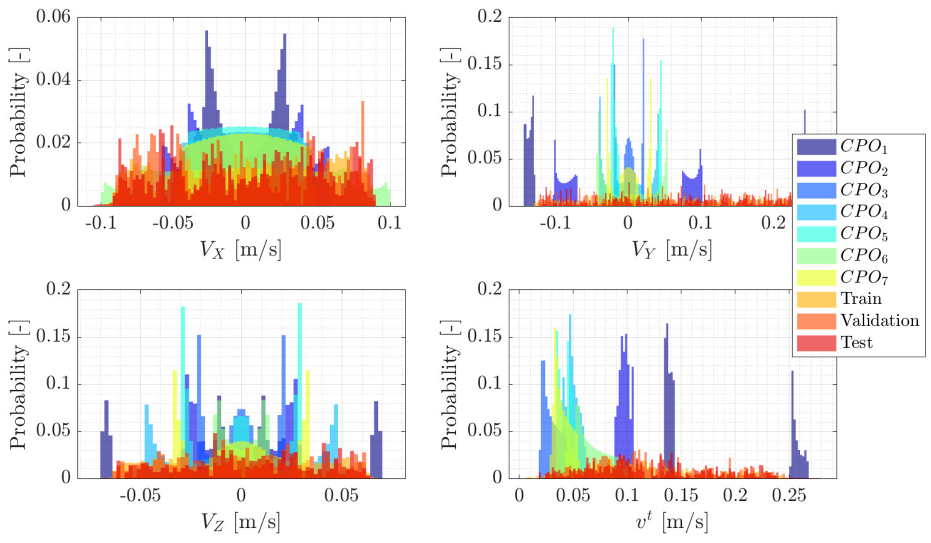


Figure A.22: Main properties of $\mathcal{DS}_6$ in velocity-phase space. The binwidth is set to *0.002* m/s and *probability* is used as normalization.

A.3 Milani datasets

A.3.1 $\mathcal{DS}_7$

This is the dataset used in [32, 178]. The dataset is a mixed-input one, constituted both by images and feature vectors. Its geometrical properties are summarized in Table A.18. The camera poses are illustrated in Figure A.23 while dataset statistics are illustrated in Figure A.24.

Table A.18: Properties of $\mathcal{DS}_7$.

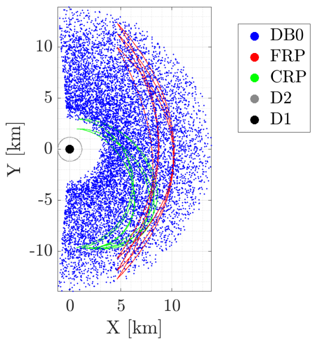
Parameter	$\mathcal{DS}_7^{DB_0}$	$\mathcal{DS}_7^{FRP}$	$\mathcal{DS}_7^{CRP}$
Number of images	10000	12102	16020
Range [km]	[4.0, 14.0]	[8.7, 14.0]	[2.1, 11.0]
Azimuth [deg]	[-95.0, 95.0]	[-70.0, 69.5]	[-86.0, 74.6]
Elevation [deg]	[-45.0, 45.0]	[-39.4, 33.6]	[-12.9, 16.8]
D1 is observable [%]	100	100	100
D2 is not observable [%]	19.04	11.12	14.62
D2 separated from D1 [%]	73.40	76.54	61.62
D2 close to D1 [%]	7.56	12.35	13.76

$\mathcal{DS}_7$ is divided into three subsets: $\mathcal{DS}_7^{DB_0}$, used for training and validation, $\mathcal{DS}_7^{FRP}$ and $\mathcal{DS}_7^{CRP}$, used for testing. All datasets represent imaging conditions around the Didymos binary system. $\mathcal{DS}_7^{DB_0}$ represents a statistical sample of images of the system seen from different points of view. It comprehends randomly generated points in the $\mathcal{W}$ reference frame with the properties illustrated in Table A.18. During the generation of the images, the angular position of the D2 with respect to D2 is changed randomly, constraining D2 to be tidally locked with D1. Instead, $\mathcal{DS}_7^{FRP}$ and $\mathcal{DS}_7^{CRP}$ are obtained by sampling the FRP and CRP trajectories of the Milani mission every 150 s while using the bodies ephemerides.

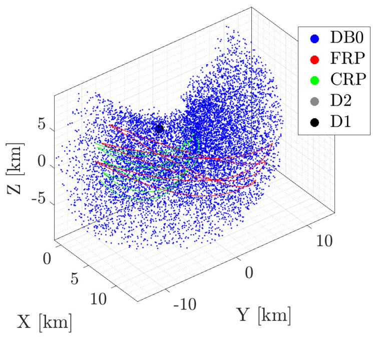
For simplicity, ideal pointing towards the CoM of D1 is assumed, and the images are obtained with the NavCam without the application of artificial noise. Moreover, the X axis of the NavCam is aligned with the equatorial plane of the binary system, assuming that the Z axis represents the boresight direction and the X and Y axes are, respectively the ones associated with the longest and shortest size of the sensor. The shape models of D1 and D2 are enhanced versions of the baseline models from [138], processed with procedural changes and re-mesh to simulate roughness and albedo variations with cloud and Voronoi patterns. A 10 m crater is also added on D2.

Due to the geometrical configuration of D1, D2, and Milani, several occultation phenomena can happen that act as a disturbance in the images.

Finally, the images of $\mathcal{DS}_7$ are also used to extract 12-component feature vectors from geometrical quantities extracted from the images. These components



(a) XY view.



(b) 3D view.

Figure A.23: Visualization of $\mathcal{DS}_7$ in position phase-space in the $\mathcal{W}$ reference frame. (a) top-view and (b) 3D view.

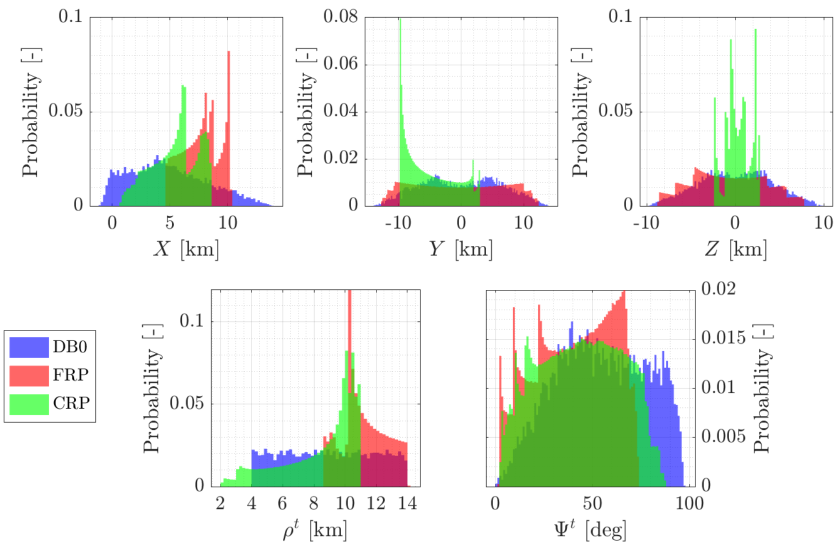


Figure A.24: Main geometric properties of $\mathcal{DS}_7$. The binwidth is set to *0.2* km and *1* deg, and *probability* is used as normalization.

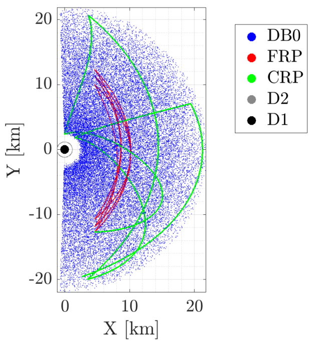
correspond to the bounding box $\Gamma = [\,\Gamma_u,\, \Gamma_v,\, \Gamma_w,\, \Gamma_h\,]$ represented by its upper-left corner coordinates and its width and height, the eccentricity e, minor and major axis lengths, respectively δ_m and δ_M, and orientation θ of the ellipse fitted to the blobs of pixel associated with D1 with the same second order moment, the perimeter p, area A of the blobs of pixels associated to D1, the circularity c, computed as $\frac{4\pi A}{p^2}$, the equivalent diameter δ_{eq} of the circle which has the same area of the blob of pixels, and the coordinate of the CoB of the blobs of pixels associated with D1. These quantities can be used in substitution of the images to represent them.

A.3.2 $\mathcal{DS}_8$

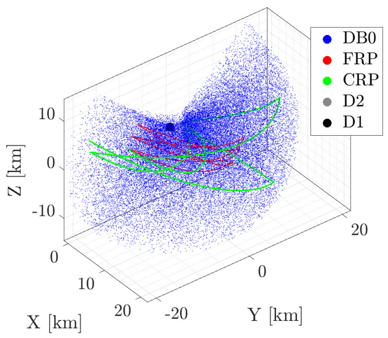
This is the dataset used in [158]. The dataset is constituted by image-label pairs. Its geometrical properties are illustrated in Table A.19 while the datasets statistics and camera poses are represented in Figure A.26 and Figure A.25. The dataset is generated with the same setup of $\mathcal{DS}_7$ illustrated in Section A.3.1, the main difference being an update in the CRP trajectory.

Table A.19: Properties of $\mathcal{DS}_8$.

Parameter	$\mathcal{DS}_8^{DB_0}$	$\mathcal{DS}_8^{FRP}$	$\mathcal{DS}_8^{CRP}$
Number of images	36400	12102	20164
Range [km]	[3.0, 22.0]	[8.7, 14.0]	[2.4, 21.3]
Azimuth [deg]	[-92.0, 92.0]	[-70.0, 69.5]	[-82.0, 90.5]
Elevation [deg]	[-42.0, 42.0]	[-39.4, 33.6]	[-10.0, 20.0]
D1 is observable [%]	100	100	100
D2 is not observable [%]	19.15	11.54	17.21
D2 separated from D1 [%]	72.41	76.17	68.75
D2 close to D1 [%]	8.45	12.29	14.04



(a) Top-view.



(b) 3D view.

Figure A.25: Visualization of $\mathcal{DS}_8$ in position phase-space in the $\mathcal{W}$ reference frame. (a) top-view and (b) 3D view.

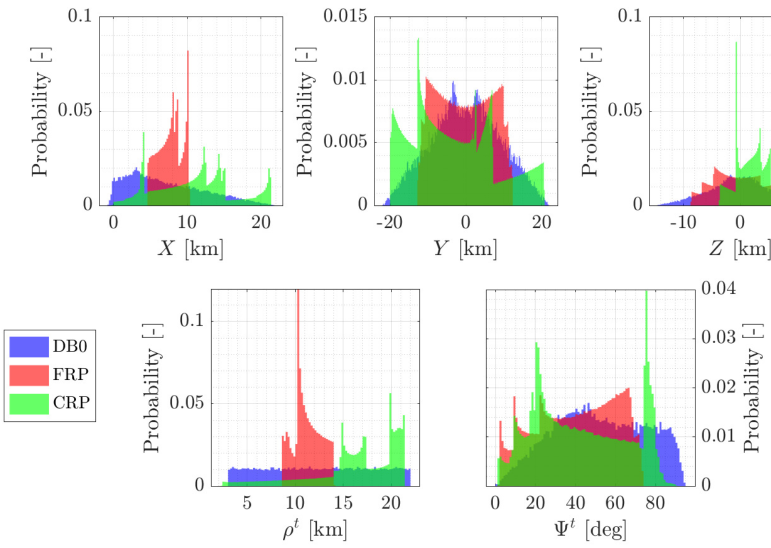


Figure A.26: Main geometric properties of $\mathcal{DS}_8$. The binwidth is set to *0.2* km and *1* deg, and *probability* is used as normalization.

A.3.3 $\mathcal{DS}_9$

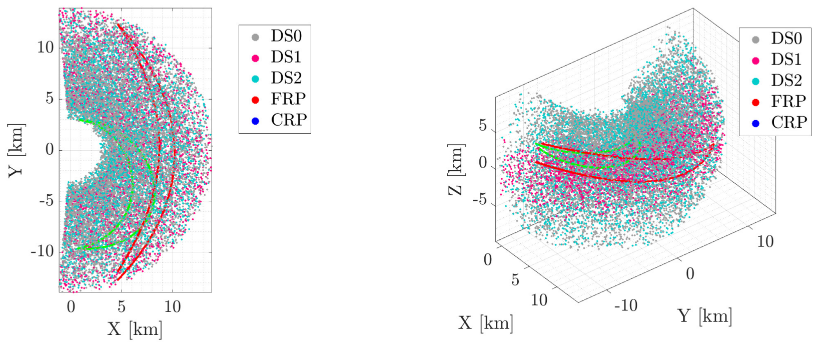
This is the dataset used in [119]. The dataset is constituted by image-label pairs. Its geometrical properties are illustrated in Table A.20 while the datasets statistics and camera poses are represented in Figure A.28 and Figure A.27.

Table A.20: Properties of $\mathcal{DS}_9$.

Parameter	$\mathcal{DS}_9^{DS0}$	$\mathcal{DS}_9^{DS1}$	$\mathcal{DS}_9^{DS2}$	$\mathcal{DS}_9^{FRP}$	$\mathcal{DS}_9^{CRP}$
Number of images	10000	5000	2840	4032	4032
Range [km]	[4.0, 14.0]	[4.0, 14.0]	[4.0, 14.0]	[8.7, 14.0]	[2.1, 11.0]
Azimuth [deg]	[-95.0, 95.0]	[-95.0, 95.0]	[-95.0, 95.0]	[-70.0, 69.5]	[-86.0, 74.6]
Elevation [deg]	[-45.0, 45.0]	[-18.0, 18.0]	[-45.0, 45.0]	[-39.4, 33.6]	[-12.9, 16.8]

As of $\mathcal{DS}_8$, the dataset is generated with the same setup of $\mathcal{DS}_7$ illustrated in Section A.3.1, the main difference being the camera's properties. In $\mathcal{DS}_9$, the camera is not modeled as the Milani's camera, but rather following the specification of the Basler camera mounted by default within the TinyV3RSE facility, illustrated in Section 3.2.

The camera poses of $\mathcal{DS}_9^{DS0}$, $\mathcal{DS}_9^{DS1}$, and $\mathcal{DS}_9^{DS2}$ are set randomly. $\mathcal{DS}_9^{DS0}$ is used for training, while $\mathcal{DS}_9^{DS1}$ and $\mathcal{DS}_9^{DS2}$ are used for testing. $\mathcal{DS}_9^{DS1}$ represents random conditions different than $\mathcal{DS}_9^{DS0}$ while $\mathcal{DS}_9^{DS2}$ represents challenging conditions in which D2 is overlapping partially or totally with D1. These cases are



(a) XY view. **(b)** 3D view.

Figure A.27: Visualization of $\mathcal{DS}_9$ in position phase-space in the $\mathcal{W}$ reference frame. (a) top-view and (b) 3D view.

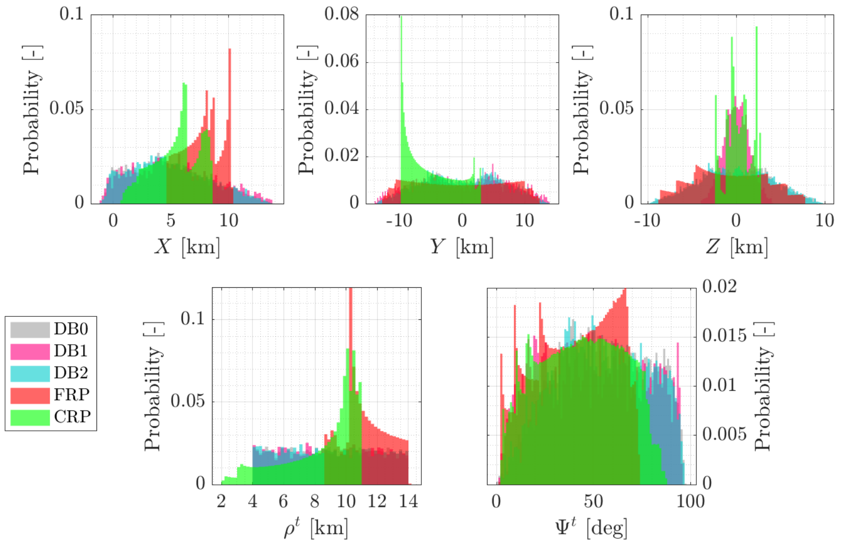


Figure A.28: Main geometric properties of $\mathcal{DS}_9$. The binwidth is set to *0.2* km and *1* deg, and *probability* is used as normalization.

explored since they tend to worsen the algorithm's performance. Finally, $\mathcal{DS}_9^{FRP}$ and $\mathcal{DS}_9^{CRP}$ are test cases sampled every *150* s from the first two arcs of the FRP and CRP of Milani. In these datasets, the only modification made to the trajectory is to slightly scale the range to compensate for the different FOV of the mission camera employed with respect to Milani's navigation camera.

Exploiting $\mathcal{DS}_9^{DS1}$, three additional datasets are generated sharing the same geometric conditions. In the first one, the dimensions of the body along all directions have been reduced by 5%. In the second set, only the z direction of the body frame has been scaled by 5%, thus increasing the oblateness of the body. Finally, in the last set, the oblateness has instead been reduced by scaling only the x and y axes of the body by 5%. These values have been selected from [138].

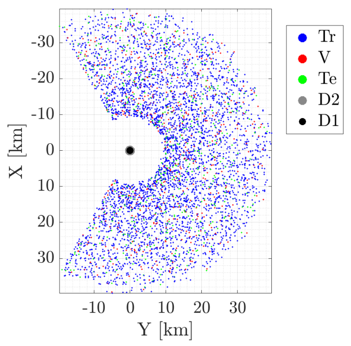
All the datasets, except $\mathcal{DS}_9^{DS0}$, have been acquired using TinyV3RSE to test the performance of the algorithms. $\mathcal{DS}_9^{DS1}$ has been acquired with different camera settings to study the effect of the exposure time and the blur on algorithm performance. The exposure time has been set to span from highly under-exposed images to highly over-exposed ones. In particular, five values have been tested: *1* ms, *4* ms, *7* ms, *10* ms, and *50* ms. Furthermore, after fixing the exposure time, images with three levels of blur have been acquired, characterized by an increasingly wider point-spread-function with σ_{blur} equal to *0.77*, *1.47* and *2.60* pixel. Samples of images of $\mathcal{DS}_9^{DS1}$ are illustrated in Figure 6.29 and Figure 6.30.

A.3.4 $\mathcal{DS}_{10}$

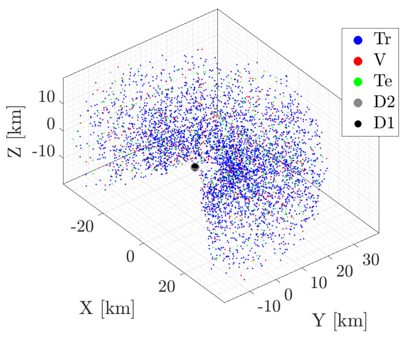
This is the dataset used in [164]. The dataset is a mixed-input one, constituted both by images and feature vectors. Its geometrical properties are summarized in Table A.21. The camera poses are illustrated in Figure A.29 while dataset statistics are illustrated in Figure A.30.

Table A.21: Properties of $\mathcal{DS}_{10}$.

Parameter	$\mathcal{DS}_{10}^{Tr}$	$\mathcal{DS}_{10}^{V}$	$\mathcal{DS}_{10}^{Te}$
Number of images	4000	500	500
Range [km]	[10, 40]	[10, 40]	[10, 40]
Azimuth [deg]	[-120, 120]	[-120, 120]	[-120, 120]
Elevation [deg]	[-30, 30]	[-30, 30]	[-30, 30]



(a) XY view.



(b) 3D view.

Figure A.29: Visualization of $\mathcal{DS}_{10}$ in position phase-space in the $\mathcal{W}$ reference frame. (a) top-view and (b) 3D view.

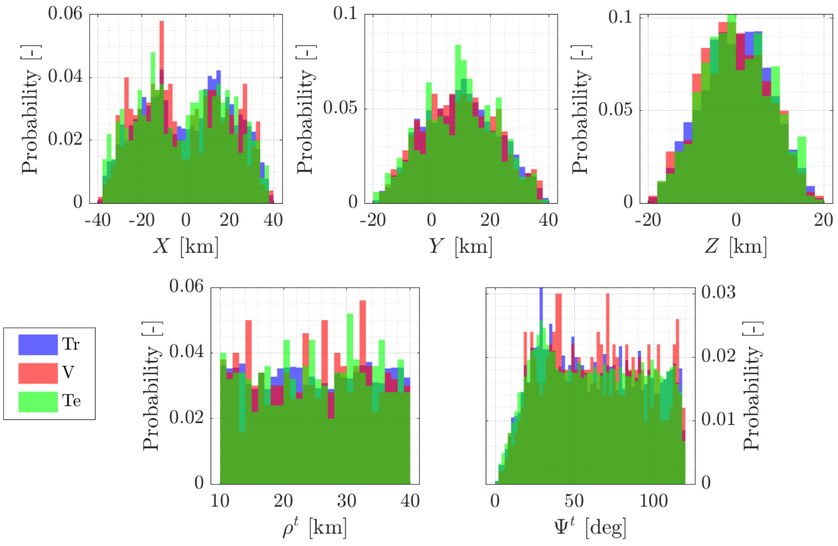


Figure A.30: Main geometric properties of $\mathcal{DS}_{10}$. The binwidth is set to *0.2* km and *1* deg, and *probability* is used as normalization.

Once again, the rendering procedure is similar to the one illustrated for $\mathcal{DS}_7$, with few substantial differences. Images are rendered in Blender using CORTO and the *Cycles* rendering engine at 1024×1024 pixels with a FOV of *5.5* deg. These conditions are representative of the onboard Hera's navigation camera. The images are also rendered with an ideal pointing towards the CoM of D1 and with a random boresight rotation. Domain randomization is applied by adding a variable artificial noise to each image and by changing the albedo and coefficients of the scattering law used at each acquisition.

The same geometric conditions for the renderings are used to generate different versions of the $\mathcal{DS}_{10}$ dataset with different scaling of the Didymos model across the z-axis s_z.

To generate the feature vector associated with each image, the images of the $\mathcal{DS}_{10}$ dataset with $s_z = 0.78 s_0$ are passed through the IP of Milani. These steps generate intermediate geometrical features that are guaranteed to be generated onboard the CubeSat. Each image is then associated with 14 components feature vector. The components can be divided into three main groups: associated directly with the blob of pixels of D1 (f_{D1}), associated with the blob of pixels of the edge region of D1 (f_{edge}), and combined properties between f_{D1} and f_{edge} (f_{comb}).

$$f_{D1} = [\log_{10}(\nu_{area}), \log_{10}(\nu_{per}), \nu_{circ}, \nu_{ext}, \nu_e]^{D1} \tag{A.5}$$

$$f_{edge} = [\log_{10}(\nu_{area}), \log_{10}(\nu_{per}), \nu_{circ}, \nu_{ext}, \nu_e]^{edge} \tag{A.6}$$

To compose f_{D1} and f_{edge} the area (ν_{area}), perimeter (ν_{per}), circularity (ν_{circ}), extent (ν_{ext}), and eccentricity (ν_e) are put together. On the other hand, to compose f_{comb}, more complex relationships are used based on previous experience. ν_1 is evaluated as the ratio between the perimeter of the blob of pixels of D1 and the sum of the perimeters of the multiple edge regions detected in the image. ν_2 is evaluated as the ratio between the perimeter of the blob of pixels of D1 and the sum of the areas of the multiple edge regions detected in the image. ν_3 is computed as a summation over the entire image of the normalized activation map after the application of a Sobel filter to the image. Finally, ν_4 is computed as the ratio between the eccentricity of the blob of pixels of D1 and the eccentricity of the edge region. These components are then put together as follows:

$$f_{comb} = [\tanh(\nu_1), \tanh(\nu_2), \log_{10}(\nu_3), \nu_4]^{comb} \tag{A.7}$$

Finally, for each image, the feature vector of 14 elements is composed as $f_x = [f_{D1}, f_{edge}, f_{comb}]$. It is emphasized that the full $\mathcal{DS}_{10}$ datasets with $s_z = 0.78 s_0$ and $s_z = s_0$ are made publicly available for the interested readers at the following link together with the values illustrated in Section 6.6.3.3, and the expansion coefficients for the aPC and PCE: https://zenodo.org/record/7962714

References

[1] Quadrelli MB et al. "Guidance, Navigation, and Control Technology Assessment for Future Planetary Science Missions". In: *Journal of Guidance, Control, and Dynamics* 38.7 (July 2015), pp. 1165–1186. DOI: 10.2514/1.g000525.

[2] Walker R, Binns D, Bramanti C, et al. "Deep-space CubeSats: thinking inside the box". In: *Astronomy & Geophysics* 59.5 (2018), pp. 5–24. DOI: 10.1093/astrogeo/aty237.

[3] Poghosyan A and Golkar A. "CubeSat evolution: Analyzing CubeSat capabilities for conducting science missions". In: *Progress in Aerospace Sciences* 88 (2017), pp. 59–83. DOI: 10.1016/j.paerosci.2016.11.002.

[4] Bandyopadhyay S, Foust R, Subramanian GP, Chung SJ, and Hadaegh FY. "Review of formation flying and constellation missions using nanosatellites". In: *Journal of Spacecraft and Rockets* 53.3 (2016), pp. 567–578. DOI: 10.2514/1.A33291.

[5] Kalita H, Asphaug E, Schwartz S, and Thangavelautham J. *Network of Nano-Landers for In-Situ Characterization of Asteroid Impact Studies*. 2017. arXiv: 1709.02885 [cs.RO].

[6] Turan E, Speretta S, and Gill E. "Autonomous navigation for deep space small satellites: Scientific and technological advances". In: *Acta Astronautica* 193 (2022), pp. 56–74. ISSN: 0094-5765. DOI: 10.1016/j.actaastro.2021.12.030.

[7] Cervone A et al. "LUMIO: A CubeSat for observing and characterizing micro-meteoroid impacts on the Lunar far side". In: *Acta Astronautica* 195 (2022), pp. 309–317. DOI: 10.1016/j.actaastro.2022.03.032.

[8] Serio GF, Manara A, Sicoli P, and Bottke WF. *Giuseppe Piazzi and the discovery of Ceres*. University of Arizona Press, 2002.

[9] Lissauer J and Pater I de. *Fundamental Planetary Science: Physics, Chemistry and Habitability*. Cambridge University Press, 2013. ISBN: 9781107354616. URL: https://books.google.co.id/books?id=0iggAwAAQBAJ.

[10] Olson RJM. *Fire and ice: a history of comets in art*. 1985.

[11] Michel P, DeMeo FE, and Bottke WF. *Asteroids IV*. University of Arizona Press, 2015.

[12] Prockter L et al. "The NEAR shoemaker mission to asteroid 433 eros". In: *Acta Astronautica* 51.1 (2002), pp. 491–500. ISSN: 0094-5765. DOI: 10.1016/S0094-5765(02)00098-X.

[13] Yoshikawa M, Kawaguchi J, Fujiwara A, and Tsuchiyama A. "Hayabusa sample return mission". In: *Asteroids IV* 1 (2015), pp. 397–418. DOI: 10.2458/azu_uapress_9780816532131-ch021.

[14] Russell CT and Raymond CA. "The Dawn Mission to Vesta and Ceres". In: *The Dawn Mission to Minor Planets 4 Vesta and 1 Ceres*. Ed. by Russell C and Raymond C. New York, NY: Springer New York, 2012, pp. 3–23. ISBN: 978-1-4614-4903-4. DOI: 10.1007/978-1-4614-4903-4_2.

[15] Glassmeier KH, Boehnhardt H, Koschny D, Kührt E, and Richter I. "The Rosetta mission: flying towards the origin of the solar system". In: *Space Science Reviews* 128 (2007), pp. 1–21. DOI: 10.1007/s11214-006-9140-8.

[16] Watanabe Si et al. "Hayabusa2 Mission verview". In: *Space Science Reviews* 208.1 (2017), pp. 3–16. DOI: 10.1007/s11214-017-0377-1.

[17] Lauretta DS et al. "OSIRIS-REx: Sample Return from Asteroid (101955) Bennu". In: *Space Science Reviews* 212.1 (2017), pp. 925–984. DOI: 10.1007/s11214-017-0405-1.

[18] A'Hearn MF et al. "Deep impact: excavating comet Tempel 1". In: *science* 310.5746 (2005), pp. 258–264. DOI: 10.1126/science.1118923.

[19] Rivkin AS et al. "The Double Asteroid Redirection Test (DART): Planetary Defense Investigations and Requirements". In: *The Planetary Science Journal* 2.5 (Aug. 2021), p. 173. DOI: 10.3847/PSJ/ac063e/meta.

[20] *Defending Planet Earth: Near-Earth-Object Surveys and Hazard Mitigation Strategies*. National Academies Press, June 2010, pp. 85–87. DOI: 10.17226/12842.

[21] Morris T. *Computer vision and image processing*. Palgrave Macmillan Ltd, 2004.

[22] Szeliski R. *Computer Vision*. 2nd. Springer International Publishing, 2022. Chap. 6, pp. 307–316. DOI: 10.1007/978-3-030-34372-9.

[23] Boden MA. *Mind as machine: A history of cognitive science*. Oxford University Press, 2008.

[24] Song J, Rondao D, and Aouf N. "Deep learning-based spacecraft relative navigation methods: A survey". In: *Acta Astronautica* 191 (2022), pp. 22–40. DOI: 10.1016/j.actaastro.2021.10.025.

[25] Izzo D, Märtens M, and Pan B. "A Survey on Artificial Intelligence Trends in Spacecraft Guidance Dynamics and Control". In: *Astrodynamics* 3.4 (July 2019), pp. 287–299. DOI: 10.1007/s42064-018-0053-6.

[26] Nesnas IAD et al. "Autonomous Exploration of Small Bodies Toward Greater Autonomy for Deep Space Missions". In: *Frontiers in Robotics and AI* 8 (Nov. 2021). DOI: 10.3389/frobt.2021.650885.

[27] Gil-Fernandez J and Ortega-Hernando G. "Autonomous vision-based navigation for proximity operations around binary asteroids". In: *CEAS Space Journal* 10.2 (Feb. 2018), pp. 287–294. DOI: 10.1007/s12567-018-0197-5.

[28] Lessac-Chenen EJ et al. "Optical Navigation Operations and Preparations for the Lucy Trojan-Asteroid Mission". In: *AIAA SCITECH 2022 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2022. DOI: 10.2514/6.2022-1226.

[29] Bekker D, Smith R, and Tran MQ. "Guiding DART to Impact — the FPGA SoC Design of the DRACO Image Processing Pipeline". In: *2021 IEEE Space Computing Conference (SCC)*. IEEE, Aug. 2021. DOI: 10.1109/scc49971.2021.00020.

[30] Buonagura C, Pugliatti M, and Topputo F. "Image Processing Robustness Assessment of Small-Body Shapes". In: *The Journal of the Astronautical Sciences* 69.6 (Nov. 2022), pp. 1744–1765. DOI: 10.1007/s40295-022-00348-6.

[31] Bhaskaran S, Riedel J, and Synnott S. "Autonomous nucleus tracking for comet/asteroid encounters: the Stardust example". In: *1998 IEEE Aerospace Conference Proceedings (Cat. No.98TH8339)*. IEEE. DOI: 10.1109/aero.1998.687921.

[32] Pugliatti M, Franzese V, and Topputo F. "Data-Driven Image Processing for Onboard Optical Navigation Around a Binary Asteroid". In: *Journal of Spacecraft and Rockets* 59.3 (2022), pp. 943–959. DOI: 10.2514/1.A35213.

[33] Wright C, Liounis AJ, and Ashman B. *Optical navigation algorithm performance*. 1st Annual RPI Workshop on Image-Based Modeling and Navigation for Space Applications, Troy, NY. 2018.

[34] Christian JA and Robinson SB. "Noniterative Horizon-Based Optical Navigation by Cholesky Factorization". In: *Journal of Guidance, Control, and Dynamics* 39.12 (Dec. 2016), pp. 2757–2765. DOI: 10.2514/1.g000539.

[35] Liounis AJ. *Limb Based Optical Navigation for Irregular Bodies*. 1st Annual RPI Workshop on Image-Based Modeling and Navigation for Space Applications, Troy, NY. 2018.

[36] Christian JA. "Optical Navigation Using Planet's Centroid and Apparent Diameter in Image". In: *Journal of Guidance, Control, and Dynamics* 38.2 (Feb. 2015), pp. 192–204. DOI: 10.2514/1.g000872.

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

[38] Teil T, Schaub H, and Kubitschek D. "Centroid and Apparent Diameter Optical Navigation on Mars Orbit". In: *Journal of Spacecraft and Rockets* 58.4 (July 2021), pp. 1107–1119. DOI: 10.2514/1.a34815.

[39] Owen WMJ. "Methods of optical navigation". In: *AAS Spaceflight Mechanics Conference, New Orleans, Louisiana*. 2011, pp. 1–19.

[40] Pellacani A, Graziano M, Fittock M, Gil J, and Carnelli I. "HERA vision based GNC and autonomy". In: (2019). DOI: 10.13009/EUCASS2019-39.

[41] Qian W, Wei Z, De X, and Xiaoyan M. "Model-based line-of-sight detection of an irregular celestial body for autonomous optical navigation". In: *2015 34th Chinese Control Conference (CCC)*. 2015, pp. 5527–5532. DOI: 10.1109/ChiCC.2015.7260503.

[42] Baker DA and McMahon JW. "Limb-based shape modeling and localization for autonomous navigation around small bodies". In: *2020 Astrodynamic Specialist Conference, Lake Tahoe*. AAS/AIAA, Aug. 2020.

[43] Kaluthantrige A, Feng J, and Gil-Fernández J. "CNN-based image processing algorithm for autonomous optical navigation of Hera mission to the binary asteroid Didymos". In: *Acta Astronautica* 211 (May 2023), pp. 60–75. ISSN: 0094-5765. DOI: 10.1016/j.actaastro.2023.05.029.

[44] Lopez AE, Ghiglino P, and Sanjurjo-rivo M. "Churinet - Applying Deep Learning for Minor Bodies Optical Navigation". In: *IEEE Transactions on Aerospace and Electronic Systems* (2022), pp. 1–14. DOI: 10.1109/TAES.2022.3227497.

[45] Hijden L van der. "Autonomous Navigation around Asteroids using Convolutional Neural Networks". TU Delft, 2022.

[46] Sharma S, Beierle C, and D'Amico S. "Pose estimation for non-cooperative spacecraft rendezvous using convolutional neural networks". In: *2018 IEEE Aerospace Conference*. IEEE, Mar. 2018. DOI: 10.1109/aero.2018.8396425.

[47] Teil T, Bateman S, and Schaub H. "Autonomous On-orbit Optical Navigation Techniques For Robust Pose-Estimation". In: *Advances in the Astronautical Sciences AAS Guidance, Navigation, and Control* 172 (2020).

[48] Scorsoglio A et al. "Image-Based Deep Reinforcement Meta-Learning for Autonomous Lunar Landing". In: *Journal of Spacecraft and Rockets* 59.1 (2022), pp. 153–165. DOI: 10.2514/1.A35072.

[49] Gaudet B, Linares R, and Furfaro R. "Deep reinforcement learning for six degree-of-freedom planetary landing". In: *Advances in Space Research* 65.7 (Apr. 2020), pp. 1723–1741. DOI: 10.1016/j.asr.2019.12.030.

[50] Furfaro R and Law AM. "Relative optical navigation around small bodies via extreme learning machines". In: *AAS/AIAA Astrodynamics Specialist Conference 2015*. Vol. 156. Univelt Inc., 2016, pp. 1959–1978. ISBN: 9780877036296.

[51] Panicucci P. "Autonomous vision-based navigation and shape reconstruction of an unknown asteroid during approach phase". PhD thesis. Institut Supérieur de l'Aéronautique et de l'Espace, 2021.

[52] Jarvis B et al. "3D Shape Reconstruction of Small Bodies From Sparse Features". In: *IEEE Robotics and Automation Letters* 6.4 (2021), pp. 7089–7096. DOI: 10.1109/LRA.2021.3097273.

[53] Wagstaff KL, Thompson DR, Bue BD, and Fuchs TJ. "Autonomous Real-time Detection of Plumes and Jets from Moons and Comets". In: *The Astrophysical Journal* 794.1, 43 (Oct. 2014), p. 43. DOI: 10.1088/0004-637X/794/1/43.

[54] Fuchs TJ et al. "Enhanced flyby science with onboard computer vision: Tracking and surface feature detection at small bodies". In: *Earth and Space Science* 2.10 (Oct. 2015), pp. 417–434. DOI: 10.1002/2014ea000042.

[55] Iiyama K, Tomita K, Bhavi A. Jagatiaz and TN, and Ho K. "Deep Reinforcement Learning for safe landing site selection with concurrent consideration of divert maneuvers". In: *AAS/AIAA Astrodynamics Specialist Conference 2020*. Vol. 175. Univelt Inc., San Diego, CA, Aug. 2020, pp. 111–130. ISBN: 9780877036753.

[56] Peñarroya P, Centuori S, Sanjurjo M, and Hermosín P. "A LiDAR-less approach to autonomous hazard detection and avoidance systems based on semantic segmentation". In: *Celestial Mechanics and Dynamical Astronomy* 135.3 (May 2023). DOI: 10.1007/s10569-023-10140-9.

[57] Tomita K, Skinner KA, and Ho K. "Bayesian Deep Learning for Segmentation for Autonomous Safe Planetary Landing". In: *Journal of Spacecraft and Rockets* 59.6 (Nov. 2022), pp. 1800–1808. DOI: 10.2514/1.a35104.

[58] Caroselli E, Belien F, Falke A, Curti F, and Forstner R. "Deep learning-based passive hazard detection for asteroid landing in unexplored environment". In: *44th AAS GN&C conference, Colorado, Breckenridge*. AAS 22-044. Feb. 2022, pp. 1–16.

[59] Palafox LF, Hamilton CW, Scheidt SP, and Alvarez AM. "Automated detection of geological landforms on Mars using Convolutional Neural Networks". In: *Computers & Geosciences* 101 (Apr. 2017), pp. 48–56. DOI: 10.1016/j.cageo.2016.12.015.

[60] Santayana RP de and Lauer M. "Optical measurements for rosetta navigation near the comet". In: *Proceedings of the 25th International Symposium on Space Flight Dynamics (ISSFD), Munich*. 2015.

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

[62] Mario C, Debrunner C, et al. "Robustness and performance impacts of optical-based feature tracking to OSIRIS-REx asteroid sample collection mission". In: *39th Annual AAS Guidance and Control Conference*. 2015, pp. 513–525.

[63] Lucas BD and Kanade. T. "An Iterative Image Registration Technique with an Application to Stereo Vision". In: *Proceedings of Imaging Understanding Workshop*. 1981, pp. 121–130.

[64] Ogawa N et al. "Image-based autonomous navigation of Hayabusa2 using artificial landmarks: The design and brief in-flight results of the first landing on asteroid Ryugu". In: *Astrodynamics* 4.2 (June 2020), pp. 89–103. DOI: 10.1007/s42064-020-0070-0.

[65] Villa J, Mcmahon J, Hockman B, and Nesnas I. "Autonomous Navigation and Dense Shape Reconstruction Using Stereophotogrammetry at Small Celestial Bodies". In: *44th AAS GN&C conference, Colorado, Breckenridge*. AAS 22-044. Feb. 2022, pp. 1–23.

[66] Dor M, Driver T, Getzandanner K, and Tsiotras P. *AstroSLAM: Autonomous Monocular Navigation in the Vicinity of a Celestial Small Body – Theory and Experiments*. 2022. arXiv: 2212.00350 [cs.RO].

[67] Dor M, Skinner KA, Driver T, and Tsiotras P. "Visual SLAM for asteroid relative navigation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 2066–2075.

[68] Driver T and Tsiotras P. "Efficient Feature Description for Small Body Relative Navigation using Binary Convolutional Neural Networks". In: *arXiv preprint* (2023). DOI: 10.48550/arXiv.2304.04985.

[69] Driver T, Skinner KA, Dor M, and Tsiotras P. "AstroVision: Towards Autonomous Feature Detection and Description for Missions to Small Bodies Using Deep Learning". In: *Acta Astronautica* 210 (Sept. 2023), pp. 393–410. DOI: 10.1016/j.actaastro.2023.01.009.

[70] Christian JA, Derksen H, and Watkins R. "Lunar Crater Identification in Digital Images". In: *The Journal of the Astronautical Sciences* 68.4 (Oct. 2021), pp. 1056–1144. DOI: 10.1007/s40295-021-00287-8.

[71] Mancini P, Cannici M, and Matteucci M. "Deep learning for asteroids autonomous terrain relative navigation". In: *Advances in Space Research* 71.9 (2023). Application of Artificial Intelligence in Tracking Control and Synchronization of Spacecraft, pp. 3748–3760. ISSN: 0273-1177. DOI: 10.1016/j.asr.2022.04.020.

[72] Maass B, Woicke S, Oliveira WM, Razgus B, and Krüger H. "Crater Navigation System for Autonomous Precision Landing on the Moon". In: *Journal of Guidance, Control, and Dynamics* 43.8 (Aug. 2020), pp. 1414–1431. DOI: 10.2514/1.g004850.

[73] Rosenblatt F. "The perceptron: a probabilistic model for information storage and organization in the brain". In: *Psychological review* 65.6 (1958), p. 386. DOI: 10.1037/h0042519.

[74] Teil TF. "Optical Navigation using Near Celestial Bodies for Spacecraft Autonomy". PhD thesis. University of Colorado Boulder, 2020.

[75] Rumelhart DE, Hinton GE, Williams RJ, et al. *Learning internal representations by error propagation*. 1985.

[76] Smith LN. *A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay*. 2018. DOI: 10.48550/ARXIV.1803.09820.

[77] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, and Salakhutdinov R. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[78] LeCun Y, Bottou L, Bengio Y, and Haffner P. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[79] Krizhevsky A, Ilya S, and E. HG. "ImageNet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* (2012), pp. 1097–1105. DOI: 10.1145/3065386.

[80] Huang GB, Zhu QY, and Siew CK. "Extreme learning machine: Theory and applications". In: *Neurocomputing* 70.1-3 (Dec. 2006), pp. 489–501. DOI: 10.1016/j.neucom.2005.12.126.

[81] Huang GB. “An Insight into Extreme Learning Machines: Random Neurons, Random Features and Kernels”. In: *Cognitive Computation* 6.3 (Apr. 2014), pp. 376–390. DOI: 10.1007/s12559-014-9255-2.

[82] Huang G, Huang GB, Song S, and You K. “Trends in extreme learning machines: A review”. In: *Neural Networks* 61 (Jan. 2015), pp. 32–48. DOI: 10.1016/j.neunet.2014.10.001.

[83] Saxe AM et al. “On Random Weights and Unsupervised Feature Learning”. In: *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*. Ed. by Getoor L and Scheffer T. Omnipress, 2011, pp. 1089–1096.

[84] Huang GB, Bai Z, Kasun LLC, and Vong CM. “Local Receptive Fields Based Extreme Learning Machine”. In: *IEEE Computational Intelligence Magazine* 10.2 (May 2015), pp. 18–29. DOI: 10.1109/mci.2015.2405316.

[85] Rodrigues IR, Silva Neto SR da, Kelner J, Sadok D, and Endo PT. “Convolutional Extreme Learning Machines: A Systematic Review”. In: *Informatics* 8.2 (May 2021), p. 33. DOI: 10.3390/informatics8020033.

[86] Goodfellow I, Bengio Y, and Courville A. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[87] Hochreiter S and Schmidhuber J. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[88] Martin I, Dunstan M, and Gestido MS. “Planetary surface image generation for testing future space missions with PANGU”. In: *2nd RPI Space Imaging Workshop*. 2019, pp. 1–13.

[89] Martin I and Dunstan M. *PANGU v6: Planet and Asteroid Natural Scene Generation Utility*. 2021.

[90] Parkes S, Martin I, Dunstan M, and Matthews D. “Planet surface simulation with PANGU”. In: *Space ops 2004 conference*. 2004, p. 389.

[91] Lebreton J et al. “Image simulation for space applications with the SurRender software”. In: *11th International ESA Conference on Guidance, Navigation & Control Systems, 22 - 25 June*. 2021, pp. 1–16.

[92] Pajusalu M et al. “SISPO: Space Imaging Simulator for Proximity Operations”. In: *PloS one* 17.3 (2022), e0263882. DOI: 10.1371/journal.pone.0263882.

[93] Iakubivskyi I et al. “Aspects of nanospacecraft design for main-belt sailing voyage”. In: *Advances in Space Research* 67.9 (2021). Solar Sailing: Concepts, Technology, and Missions II, pp. 2957–2980. ISSN: 0273-1177. DOI: 10.1016/j.asr.2020.07.023.

[94] Snodgrass C and Jones GH. “The European Space Agency’s Comet Interceptor lies in wait”. In: *Nature Communications* 10.1 (Nov. 2019). DOI: 10.1038/s41467-019-13470-1.

[95] Villa J et al. “Optical navigation for autonomous approach of small unknown bodies”. In: *43rd Annual AAS Guidance, Navigation & Control Conference*. Vol. 30. 2019, pp. 1–3.

[96] Villa J, Mcmahon J, and Nesnas I. “Image Rendering and Terrain Generation of Planetary Surfaces Using Source-Available Tools”. In: *46th Annual AAS Guidance, Navigation & Control Conference*. Feb. 2023, pp. 1–24.

[97] Peñarroya P, Centuori S, and Hermosín P. “AstroSim: A GNC simulation tool for small body environments”. In: *AIAA SCITECH 2022 Forum*. 2022, p. 2355.

[98] Panicucci P and Topputo F. “The TinyV3RSE Hardware-in-the-Loop Vision-Based Navigation Facility”. In: *Sensors* 22.23 (2022). ISSN: 1424-8220. DOI: 10.3390/s22239333.

[99] Pugliatti M, Franzese V, Panicucci P, and Topputo F. “TINYV3RSE: The DART Vision-Based Navigation Test-bench”. In: *AIAA Scitech 2022 Forum*. 2022, p. 1193. DOI: 10.2514/6.2022-1193.

[100] Robbins SJ. “A New Global Database of Lunar Impact Craters >1–2 km: 1. Crater Locations and Sizes, Comparisons With Published Databases, and Global Analysis”. In: *Journal of Geophysical Research: Planets* 124.4 (2019), pp. 871–892. DOI: 10.1029/2018JE005592.

[101] Mattia P and Maestrini M. “Small-Body Segmentation Based on Morphological Features with a U-Shaped Network Architecture”. In: *Journal of Spacecraft and Rockets* 59.6 (2022), pp. 1821–1835. DOI: 10.2514/1.A35447.

[102] Buonagura C, Pugliatti M, and Topputo F. “Procedural Minor Body Generator Tool for Data-Driven Optical Navigation Methods”. In: *6th CEAS Specialist Conference on Guidance, Navigation and Control-EuroGNC*. 2022.

[103] Pugliatti M and Topputo F. “Small-body shape recognition with convolutional neural network and comparison with explicit features based method”. In: *Advances in the astronautical sciences*. Vol. 175. Univelt, Aug. 2021, pp. 2539–2258.

[104] Pugliatti M and Topputo F. “Boulders identification on small bodies under varying illumination conditions”. In: *3rd Space Imaging Workshop, Georgia, Atlanta*. Oct. 2022.

[105] Christian JA. “Optical Navigation for a Spacecraft in a Planetary System”. PhD thesis. The University of Texas at Austin, 2010.

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

[107] Christian JA. "A Tutorial on Horizon-Based Optical Navigation and Attitude Determination With Space Imaging Systems". In: *IEEE Access* 9 (2021), pp. 19819–19853. DOI: 10.1109/ACCESS.2021.3051914.

[108] Kisantal M et al. "Satellite Pose Estimation Challenge: Dataset, Competition Design, and Results". In: *IEEE Transactions on Aerospace and Electronic Systems* 56.5 (2020), pp. 4083–4098. DOI: 10.1109/TAES.2020.2989063.

[109] Piccinin M. "Spacecraft relative navigation with electro-optical sensors around uncooperative targets". PhD thesis. Politecnico di Milano, 2023.

[110] Dunstan M and Martin I. *Planet and Asteroid Natural Scene Generation Utility*. 2021.

[111] Lewis JP. "Fast normalized cross-correlation". In: *Vision Interface* 95 (1995), p. 120.

[112] Chai T and Draxler RR. "Root mean square error (RMSE) or mean absolute error (MAE)?–Arguments against avoiding RMSE in the literature". In: *Geoscientific model development* 7.3 (2014), pp. 1247–1250. DOI: 10.5194/gmd-7-1247-2014.

[113] Wang Z, Bovik A, Sheikh H, and Simoncelli E. "Image quality assessment: from error visibility to structural similarity". In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.

[114] Michel P et al. "The ESA Mission: Detailed Characterization of the DART Impact Outcome and of the Binary Asteroid (65803) Didymos". In: *The Planetary Science Journal* 3.7 (July 2022), p. 160. DOI: 10.3847/psj/ac6f52.

[115] Topputo F et al. "Meteoroids detection with the LUMIO lunar CubeSat". In: *Icarus* 389 (2023), p. 115213. ISSN: 0019-1035. DOI: 10.1016/j.icarus.2022.115213.

[116] Buonagura C et al. "Deep Learning for Navigation of Small Satellites About Asteroids: an Introduction to the Deepnav Project". In: *2nd International Conference on Applied Intelligence and Informatics*. Vol. 2. AII, 2022, pp. 1–20.

[117] Vattai N. "Development and validation of a horizon-based optical navigation test facility". Politecnico di Milano, 2019.

[118] Panicucci P, Pugliatti M, Franzese V, and Topputo F. "Improvements and Applications of the DART Vision-Based Navigation Test Bench TINYV3RSE". In: *44th AAS Guidance, Navigation and Control Conference*. 2022, pp. 1–19.

[119] Piccolo F, Pugliatti M, Panicucci P, and Topputo F. "Toward Verification and Validation of the Milani Image Processing Pipeline in the Hardware-In-the-loop Testbench Tinyv3rse". In: *44th AAS Guidance, Navigation and Control Conference*. 2022, pp. 1–21.

[120] Rufino G and Moccia A. "Laboratory Test System for Performance Evaluation of Advanced Star Sensors". In: *Journal of Guidance, Control, and Dynamics* 25.2 (2002), pp. 200–208. DOI: 10.2514/2.4888.

[121] Holt GN, D'Souza CN, and Saley DW. "Orion optical navigation progress toward exploration mission 1". In: *2018 Space Flight Mechanics Meeting*. 2018, p. 1978.

[122] Andrew AM. "Multiple view geometry in computer vision". In: *Kybernetes* (2001).

[123] Heikkila J and Silvén O. "A four-step camera calibration procedure with implicit image correction". In: *Proceedings of IEEE computer society conference on computer vision and pattern recognition*. IEEE. 1997, pp. 1106–1112. DOI: 10.1109/CVPR.1997.609468.

[124] Beierle C and D'Amico S. "Variable-Magnification Optical Stimulator for Training and Validation of Spaceborne Vision-Based Navigation". In: *Journal of Spacecraft and Rockets* 56.4 (2019), pp. 1060–1072. DOI: 10.2514/1.a34337.

[125] Tang Z, Gioi RG von, Monasse P, and Morel JM. "A Precision Analysis of Camera Distortion Models". In: *IEEE Transactions on Image Processing* 26.6 (2017), pp. 2694–2704. DOI: 10.1109/TIP.2017.2686001.

[126] Samaan MA, Steffes SR, and Theil S. "Star tracker real-time hardware in the loop testing using optical star simulator". In: *Spaceflight Mechanics* 140 (2011).

[127] Zhang Z. "A Flexible New Technique for Camera Calibration". In: *IEEE Transactions on pattern analysis and machine intelligence* 22.11 (2000), pp. 1330–1334. DOI: 10.1109/34.888718.

[128] Krüger H and Theil S. "Tron - hardware-in-the-loop test facility for lunar descent and landing optical navigation". In: *IFAC Proceedings Volumes* 43.15 (2010), pp. 265–270. DOI: 10.3182/20100906-5-jp-2022.00046.

[129] Zwick M, Huertas I, Gerdes L, and Ortega G. "ORGL–ESA'S test facility for approach and contact operations in orbital and planetary environments". In: *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS), Madrid, Spain*. Vol. 6. 2018.

[130] Park TH, Bosse J, and D'Amico S. "Robotic testbed for rendezvous and optical navigation: Multi-source calibration and machine learning use cases". In: *arXiv preprint* (2021). DOI: 10.48550/arXiv.2108.05529.

[131] Pellacani A, Graziano M, and Suatoni M. "Design, Development, Validation and Verification of GNC technologies". In: *8th European Conference for Aeronautics and Sciences (EUCASS)*. 2019.

[132] Park TH, Märtens M, Lecuyer G, Izzo D, and D'Amico S. "SPEED+: Next Generation Dataset for Spacecraft Pose Estimation across Domain Gap". In: *2022 IEEE Aerospace Conference (AERO)*. IEEE. 2022, pp. 1–15. DOI: 10.1109/AERO53065.2022.9843439.

[133] Tomita K, Skinner KA, and Ho K. "Uncertainty-Aware Deep Learning for Autonomous Safe Landing Site Selection". en. pre-print. 2021. DOI: 10.13140/RG.2.2.15224.98564.

[134] Pugliatti M, Maestrini M, Di Lizia P, Topputo F, et al. "On-board Small-Body Semantic Segmentation Based on Morphological Features with U-Net". In: *Advances in the Astronautical Sciences*. Vol. 176. Univelt, 2022, pp. 603–622.

[135] Sandler M, Howard A, Zhu M, Zhmoginov A, and Chen L. "MobileNetV2: Inverted Residuals and Linear Bottlenecks". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4510–4520. DOI: 10.1109/CVPR.2018.00474.

[136] Ronneberger O, Fischer P, and Brox T. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Vol. 9351. LNCS. 2015, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28.

[137] Mukhoti J and Gal Y. "Evaluating Bayesian Deep Learning Methods for Semantic Segmentation". pre-print. 2018. arXiv: 1811.12709. URL: http://arxiv.org/abs/1811.12709.

[138] ESA. *Hera Didymos Reference Model, Issue 5, Revision 4*. Issue 5, Revision 4. 2020.

[139] Goh E, Chen J, and Wilson B. "Mars Terrain Segmentation with Less Labels". In: *arXiv preprint* (2022). DOI: 10.48550/arXiv.2202.00791.

[140] Chen LC, Papandreou G, Schroff F, and Adam H. "Rethinking Atrous Convolution for Semantic Image Segmentation". In: *arXiv preprint* (2017). DOI: 10.48550/arXiv.1706.05587.

[141] Pugliatti M and Maestrini M. "A multi-scale labeled dataset for boulder segmentation and navigation on small bodies". In: *74th International Astronautical Congress, Baku, Azerbaijan*. Oct. 2023, pp. 1–8.

[142] Campbell T, Furfaro R, Linares R, and Gaylor D. "A deep learning approach for optical autonomous planetary relative terrain navigation". In: *Spaceflight Mechanics 2017*. Vol. 160. 27th AAS/AIAA Space Flight Mechanics Meeting, 2017 ; Conference date: 05-02-2017 Through 09-02-2017. Univelt Inc., 2017, pp. 3293–3302. ISBN: 9780877036371.

[143] Weyand T, Kostrikov I, and Philbin J. "PlaNet - Photo Geolocation with Convolutional Neural Networks". In: (2016), pp. 37–55. DOI: 10.1007/978-3-319-46484-83.

[144] Radosavovic I, Kosaraju RP, Girshick R, He K, and Dollar P. "Designing Network Design Spaces". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020, pp. 10428–10436.

[145] Lewis J. "Fast normalized cross-correlation". In: *Vision Interface*. Vol. 2010. 1995, pp. 120–123.

[146] Adam CD et al. "Stereophotoclinometry for OSIRIS-REx Spacecraft Navigation". In: *The Planetary Science Journal* 4.9 (Sept. 2023), p. 167. DOI: 10.3847/PSJ/ace31d.

[147] Baldini F, Harvard A, Chung SJ, Nesnas I, and Bhaskaran S. "Autonomous Small Body Mapping and Spacecraft Navigation". In: *69th International Astronautical Congress, Bremen, Germany*. Oct. 2018, pp. 1–11.

[148] Michel P, Cheng A, and Küppers M. "Asteroid Impact and Deflection Assessment (AIDA) mission: science investigation of a binary system and mitigation test". In: *European Planetary Science Congress*. Vol. 10. 2015, pp. 123–124.

[149] Cheng AF et al. "AIDA DART asteroid deflection test: Planetary defense and science objectives". In: *Planetary and Space Science* 157 (2018), pp. 104–115. ISSN: 0032-0633. DOI: 10.1016/j.pss.2018.02.015.

[150] Dotto E et al. "LICIACube-the Light Italian Cubesat for Imaging of Asteroids in support of the NASA DART mission towards asteroid (65803) Didymos". In: *Planetary and Space Science* (2021), p. 105185. DOI: 10.1016/j.pss.2021.105185.

[151] Michel P, Küppers M, and Carnelli I. "The Hera mission: European component of the ESA-NASA AIDA mission to a binary asteroid". In: *COSPAR Scientific Assembly, Pasadena, California*. 2018, pp. 1–42.

[152] Goldberg H, Karatekin O, Ritter B, et al. "The Juventas CubeSat in Support of ESA's Hera Mission to the Asteroid Didymos". In: *Small Satellite Conference*. Logan, Utah. 2019, pp. 1–7.

[153] Ferrari F, Franzese V, Pugliatti M, Giordano C, and Topputo F. "Preliminary mission profile of Hera's Milani CubeSat". In: *Advances in Space Research* 67.6 (2021), pp. 2010–2029. DOI: 10.1016/j.asr.2020.12.034.

[154] Ferrari F, Franzese V, Pugliatti M, Giordano C, and Topputo F. "Trajectory options for Hera's Milani cubesat around (65803) Didymos". In: *The Journal of the Astronautical Sciences* 68.4 (2021), pp. 973–994. DOI: 10.1007/s40295-021-00282-z.

[155] Kohout T et al. "Milani CubeSat for ESA Hera mission". In: *European Planetary Science Congress 2021, 13–24 Sep*. EPSC2021-732. 2021, pp. 1–3. DOI: 10.5194/epsc2021-732.

[156] Dirri F et al. "VISTA Instrument: A PCM-Based Sensor for Organics and Volatiles Characterization by Using Thermogravimetric Technique". In: *2018 5th IEEE International Workshop on Metrology for AeroSpace (MetroAeroSpace)*. IEEE, June 2018, pp. 150–154. DOI: 10.1109/MetroAeroSpace.2018.8453532.

[157] Pugliatti M et al. "The Milani mission: overview and architecture of the optical-based GNC system". In: *AIAA Scitech 2022 Forum*. 2022, p. 2381. DOI: 10.2514/6.2022-2381.

[158] Pugliatti M, Piccolo F, Rizza A, Franzese V, and Topputo F. "The vision-based guidance, navigation, and control system of Hera's Milani CubeSat". In: *Acta Astronautica* 210 (2023), pp. 14–28. ISSN: 0094-5765. DOI: 10.1016/j.actaastro.2023.04.047.

[159] Lundberg SM and Lee SI. "A unified approach to interpreting model predictions". In: *Proceedings of the 31st international conference on neural information processing systems*. 2017, pp. 4768–4777.

[160] Naidu S et al. "Radar observations and a physical model of binary near-Earth asteroid 65803 Didymos, target of the DART mission". In: *Icarus* 348 (2020), p. 113777. ISSN: 0019-1035. DOI: 10.1016/j.icarus.2020.113777.

[161] JHUAPL. *Design Reference Asteroid*. Issue 4, Revision 1. 2022.

[162] Robnik-Šikonja M and Kononenko I. "Theoretical and empirical analysis of ReliefF and RReliefF". In: *Machine learning* 53 (2003), pp. 23–69.

[163] Liotta M. "Ballistic capture corridors design via Polynomial Chaos Expansion". Politecnico di Milano, 2022.

[164] Pugliatti M, Giordano C, and Topputo F. "The image processing of Milani: challenges after DART impact". In: *ESA-GNC conference, Sopot, Poland*. June 2023, pp. 1–15.

[165] Pugliatti M et al. "Enhanced Vision-Based Algorithms about Small Bodies: Lessons learned from the Stardust-R experience". In: *2nd International Stardust conference, STARCON2*. Vol. 1. 2022, pp. 1–2.

[166] Giordano C et al. "The Hera Milani CubeSat mission". In: *5th COSPAR Symposium, 2023*. Apr. 2023.

[167] Bottiglieri C et al. "Trajectory design and orbit determination of Hera's Milani CubeSat". In: *Advances in the Astronautical Sciences*. Vol. 177. Univelt, 2022, pp. 81–82.

[168] Bottiglieri C et al. "Mission Analysis and Navigation Assessment for Hera's Milani CubeSat". In: *4S Symposium*. 2022, pp. 1–20.

[169] Bottiglieri C, Piccolo F, Giordano C, Ferrari F, and Topputo F. "Applied Trajectory Design for CubeSat Close-Proximity Operations around Asteroids: The Milani Case". In: *Aerospace* 10.5 (May 2023), p. 464. DOI: 10.3390/aerospace10050464.

[170] Rizza A, Piccolo F, Pugliatti M, Panicucci P, and Topputo F. "Hardware-In-the-loop Simulation Framework for CubeSats Proximity Operations: Application to the Milani Mission". In: *73rd International Astronautical Congress, Paris, France*. Oct. 2022, pp. 1–15.

[171] Peñarroya P, Pugliatti M, Centuori S, and Topputo F. "Using Blender As Contact Dynamics Engine For Cubesat Landing Simulations Within Impact Crater On Dimorphos". In: *7th IAA Planetary Defense Conference*. Vol. 2. 2021.

[172] Peñarroya P et al. "CubeSat landing simulations on small bodies using blender". In: *Advances in Space Research* (2022). ISSN: 0273-1177. DOI: 10.1016/j.asr.2022.07.044.

[173] Pugliatti M and Topputo F. "Navigation about irregular bodies through segmentation maps". In: *Advances in the Astronautical Sciences*. Vol. 176. Univelt, 2022, pp. 1169–1187.

[174] Pugliatti M, Scorsoglio A, Furfaro R, and Topputo F. "Onboard State Estimation Around Didymos with Recurrent Neural Networks and Segmentation Maps". In: *IEEE Transactions on Aerospace and Electronic Systems* XX (2023), pp. 1–14. DOI: 10.1109/TAES.2023.3288506.

[175] Pugliatti M, Piccolo F, and Topputo F. "Object Recognition Algorithms for the Didymos Binary System". In: *2nd International Conference on Applied Intelligence and Informatics*. Vol. 2. AII, 2022, pp. 1–20.

[176] Pugliatti M and Topputo F. *DOORS: Dataset fOr bOuldeRs Segmentation*. Zenodo. Sept. 2022. URL: https://zenodo.org/record/7107409.

[177] Pugliatti M and Topputo F. "Design and Application of Convolutional Architectures for Vision-Based Navigation Around Small Bodies". In: *Journal of Spacecraft and Rockets* submitted (2023), pp. 1–20.

[178] Pugliatti M et al. “Design of the on-board image processing of the Milani mission”. In: *44th AAS Guidance, Navigation and Control Conference*. 2022, pp. 1–21.

[179] Canny J. “A Computational Approach to Edge Detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6 (1986), pp. 679–698. DOI: 10.1109/TPAMI.1986.4767851.

[180] Shih F. “Image Processing and Mathematical Morphology: Fundamentals and Applications”. In: CRC Press, 2011. Chap. 3, pp. 25–35. ISBN: 9781420089448.

[181] Faraco N. “Instance segmentation for features recognition on non-cooperative resident space objects”. Politecnico di Milano, July 2020, p. 116.

[182] Russell CT and Raymond CA. “The Dawn Mission to Vesta and Ceres”. In: *The Dawn Mission to Minor Planets 4 Vesta and 1 Ceres*. New York, NY: Springer New York, 2012. Chap. The Dawn Mission to Vesta and Ceres, pp. 3–23. ISBN: 978-1-4614-4903-4. DOI: 10.1007/978-1-4614-4903-4_2. URL: https://doi.org/10.1007/978-1-4614-4903-4_2.

[183] Redmon J, Divvala S, Girshick R, and Farhadi A. “You Only Look Once: Unified, Real-Time Object Detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[184] Accomazzo A et al. “Rosetta operations at the comet”. In: *Acta Astronautica* 115 (2015), pp. 434–441. ISSN: 0094-5765. DOI: 10.1016/j.actaastro.2015.06.009.

List of acronyms

HCELM$_3$ Hybrid Convolutional Extreme Learning Machine 3.
UNet$_\mathbf{H}$ UNet Hybrid.
UNet$_\mathbf{R}$ UNet Real.
UNet$_\mathbf{S}$ UNet Synthetic.
UNet$_\mathbf{S}^\mathbf{A}$ UNet Synthetic Augmented.

ADCS Attitude Determination and Control System.
AI Artificial Intelligence.
AIDA Asteroid Impact and Deflection Assessment.
AOCS Attitude and Orbital Control System.
aPC arbitrary Polynomial Chaos.
ASI Agenzia Spaziale Italiana.
AutOpNav Autonomous Optical Navigation experiment.

BU Blender Unit.

CDR Concurrent Design Review.
CELM Convolutional Extreme Learning Machine.
CMOC CubeSat Mission Operation Centre.
CNN Convolutional Neural Network.
CoB Center of Brightness.
COB Center Of Brigthness.
CoF Center of Figure.
CoM Center of Mass.
CORTO Celestial Object Rendering TOol.
CPO Close Proximity Orbit.
CRP Close Range Phase.

D1 Didymos.

D2 Dimorphos.
DART Deep-space Astrodynamics Research & Technology.
DART Double Asteroid Redirection Test.
DFKI Deutsches Forschungszentrum für Künstliche Intelligenz.
DL Deep Learning.

EKF Extended Kalman Filter.
ELM Extreme Learning Machine.
ESA European Space Agency.
EXP EXperimental Phase.

FOV Field Of View.
FPS Frames Per Second.
FRP Far Range Phase.

GD Gradient Descent.
GNC Guidance Navigation and Control.
GPU Graphic Processing Unit.

HCELM Hybrid Convolutional Extreme Learning Machine.
HIL Hardware-In-The-Loop.

IC Initial Condition.
ICD Inter-Class Distance.
ImP Image Plane.
IMU Inertial Measurement Unit.
IoU Intersection over Union.
IP Image Processing.
ISL Inter-Satellite Link.

JAXA Japan Aerospace Exploration Agency.

KF Kalman Filter.

LASSO Least Absolute Shrinkage and Selection Operator.
Leaky ReLU Leaky Rectified Linear Unit.
LiDAR Light Detection And Ranging.
LoS Line of Sight.
LS Leas Square.
LSTM Long-Short Term Memory.
LUMIO LUnar Meteoroid Impact Observer.

MAE Mean Absolute Error.
MBGD Mini-Batch Gradient Descent.
mIoU mean Intersection over Union.
ML Machine Learning.

MONET Minor bOdy geNErator Tool.
MSE Mean Squared Error.

NASA National Aeronautics and Space Administration.
NCC Normalized Cross Correlation.
NEA near-Earth asteroid.
NEOC Navigation Experiment Operation Centre.
NFT Natural Feature Tracking.
NN Neural Network.
nReLU Normalized Rectified Linear Unit.
NRMSE Normalized Root Mean Square Error.

OD Orbit Determination.
OSL Open Shading Language.

PANGU Planetary Planet and Asteroid Natural scene Generation Utility.
PBSDF Principled Bi-directional Scatter Distribution Falloff.
PCA Principal Components Analysis.
PCE Polynomial Chaos Expansion.
PDR Preliminary Design Review.
PoE Power over Ethernet.

ReLU Rectified Linear Unit.
RIC Robotic Innovation Center.
RMSE Root Mean Squared Error.
RNN Recurrent Neural Network.
ROI Region Of Interest.
RW Reaction Wheel.

SADA Solar Array Drive Assembly.
SCCE Sparse Categorical Cross Entropy.
SGD Stochastic Gradient Descent.
SISPO Space Imaging Simulator for Proximity Operations.
SPC Stereo Photo Clinometry.
SRP Solar Radiation Pressure.
SS Sun Sensor.
SSIM Structural Similarity Index.
SSTO Sun-Stabilized Terminator Orbit.
SSWCOB Sun-Sensor Weighted Center Of Brigthness.
STM State Transition Matrix.
STR Star Tracker.

TF TensorFlow.
TinyV3RSE Tiny Versatile 3dimensional reality simulation environment.
TRL Technology Readiness Level.

UNet	U-Shaped Network.
WCOB	Weighted Center Of Brigthness.
WSCCE	Weighted Sparse Categorical Cross Entropy.
YOLO	You Only Look Once.

Colophon

This manuscript was typeset in LaTeX. The style was inspired by Dr. Gianmario Merisio's doctoral dissertation *Engineering ballistic capture for autonomous interplanetary spacecraft with limited onboard resources* (2023).