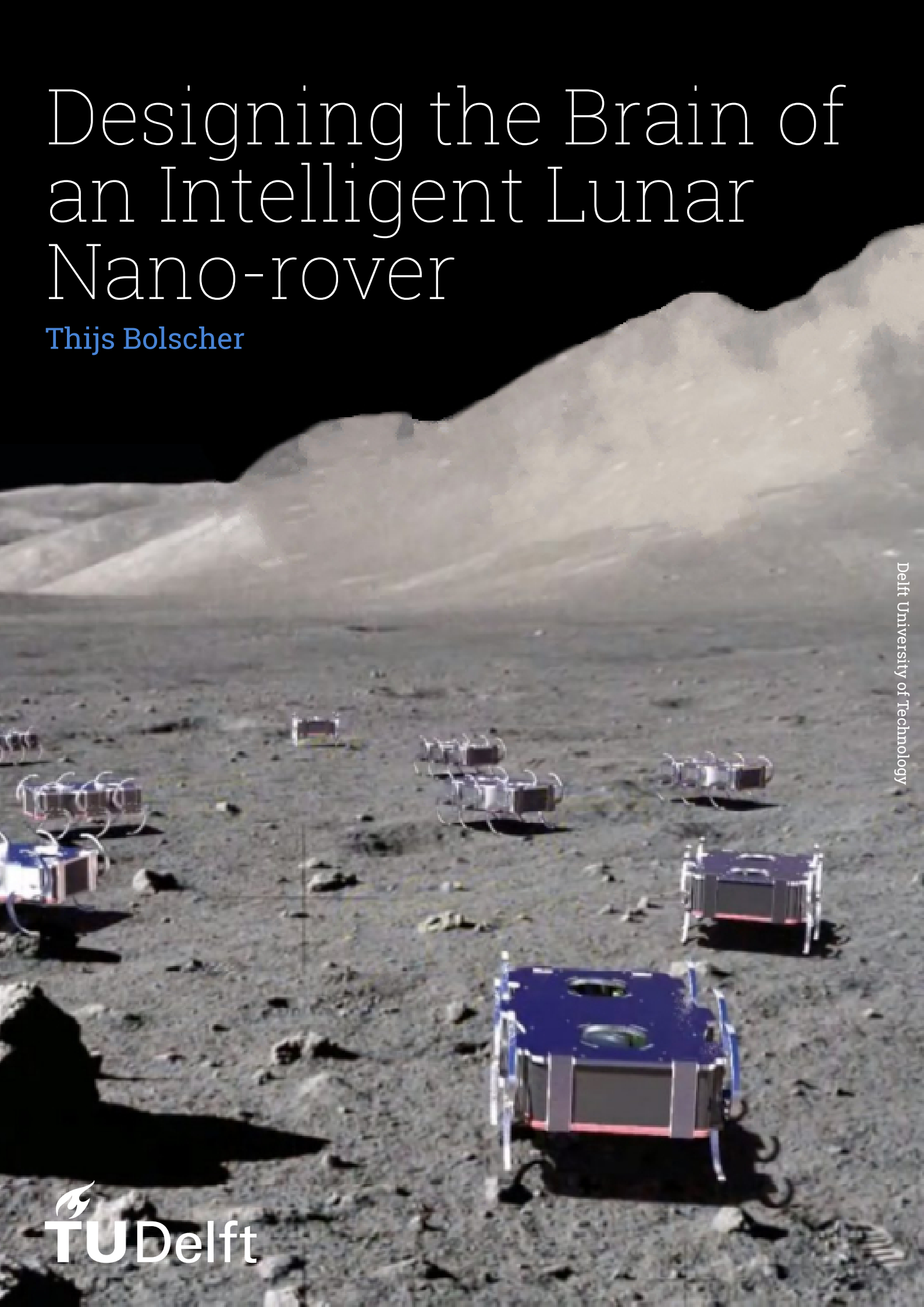
Designing the Brain of an Intelligent Lunar Nano-rover

Thijs Bolscher



Delft University of Technology

TU Delft

Designing the Brain of an Intelligent Lunar Nano-rover

by

Thijs Bolscher

to obtain thhe degree of Master of Science
at the Delft University of Technology
to be defended publicly on Wednesday January 31st, 2024 at 13:00 PM.

Student number:	4656717	
Project Duration:	March, 2023 - January, 2024	
Faculty:	Faculty of Aerospace Engineering, Delft	
Thesis committee:	Prof. Dr. J. Guo	TU Delft, Chair
	Prof. Dr. A. Menicucci	TU Delft, Supervisor
	Ir. D. Rijlaarsdam	Ubotica Technologies, Supervisor
	Prof. Dr. Ir. G.N. Gaydadjiev	TU Delft, Examiner

Cover: A swarm of Lunar Zebro rovers exploring the Lunar surface

TU Delft UBOTICA SEEING BEYOND

To those who have been there -
And to those who will return.

Preface

When I started my student life in Delft, I wanted to be challenged, and be part of the technological innovation that will change the world (and space around it). To end my student life as being part of a project that is actually on its way to the Moon is a great honor, and was a great challenge indeed. My gratitude goes out to Ubotica Technologies, for giving me the freedom to wander around in this project. I would also like to thank a couple of people who have helped me with this thesis.

First of all, thank you to my supervisors. David, for his unbelievable enthusiasm for anything techy and especially for the good conversations and friendship. Alessandra, for her guidance and her own drive to get to the Moon. One morning I walked into her office and she said; ”We are going to the Moon, Thijs!”. Well, if that doesn’t drive students, I don’t know what will...

Secondly, a special thank you to all the colleagues for their invaluable expertise, and the fun times at the office. In no particular order I would like to thank Tom, Luigi, Christopher, Keval, Tamim, Aubrey, Fintan, Maria, Leonie, Rosana, Alberto, Juan and Jose.

To my family and friends; thank you for being there for me, and making this time as fun as it was.

Finally, let’s make sure this endeavor does not end here. Let’s go to the Moon and follow Michael Collins’ advice: *”To go places and do things that have never been done before - that’s what living is all about.”*

Thijs Bolscher
Rotterdam, January 2024

Summary

The Moon, as a renewed frontier in space exploration, presents unique challenges, primarily the high costs associated with lunar missions. In response, recent initiatives have shifted focus toward deploying low-SWaP devices that are capable of conducting effective exploration on the lunar surface. The Lunar Zebro project exemplifies this trend, aiming to deploy a swarm of nano-rovers for lunar exploration.

The rise of DL technologies offers intriguing prospects for enhancing the autonomous capabilities of lunar rovers [1], improving the adaptability of the rovers to new situations. Specifically, DL-based algorithms have shown superior performance in rock segmentation tasks [2], and in the classification of hazards on the Moon [3]. The feature extraction capabilities of the Convolutional Neural Networks (CNNs) used in DL-based computer vision algorithms [4] also allow for monocular depth estimation, allowing a nanorover to detect obstacle locations from one single image [5]. The Lunar Zebro team is therefore interested in researching the integration of resource-intensive DL-based algorithms onboard these rovers. DL algorithms also allow for improved FDIR [6] allowing for the earlier discovery of faults.

Despite the computational intensity of these algorithms, advancements in COTS technologies such as GPUs, VPUs, TPUs, and FPGAs [7, 8] provide options for running these algorithms in constrained environments. The use of such technologies is being researched for applications in Earth orbit [9]. Ubotica Technologies is a company that explores the use of AI accelerators in space [10]. The company has developed the CogniSat XE2 board [11] around Intel's Myriad X VPU [12] for complex visual tasks on satellites in LEO. Their application in extraterrestrial missions remains an under-explored area of research.

The development and integration of DL-based algorithms in low Size, Weight, and Power (SWaP) rovers, along with the optimal design of On-Board Computing Architectures (OBCAs) to support these algorithms, are still areas of ongoing research. The main research question is thus posed as:

How can the OBCA for the next-generation Lunar Zebro be designed cost-effectively?

This thesis presents the design of an OBCA tailored for a lunar nano rover, using the Lunar Zebro I as a case study, taking into account the stakeholder requirement to have DL-based hazard detection on board. Employing a systematic Vee-model approach to systems engineering, three design alternatives are explored: a Single-Board Computer (SBC) with Central Processing Unit (CPU), a board with a CPU and Field Programmable Gate Array (FPGA), and an SBC with a Vision Processing Unit (VPU). Furthermore, the design of the data bus, software cycles and radiation tolerance strategies are discussed. The Xiphos Q7S, intended for the first-generation rover, and the Cognisat XE2 with the Myriad X VPU accelerator, are analyzed and tested for potential use.
For this testing procedure, a carefully selected semantic segmentation CNN, MultiResUNet [4], is trained on a Martian rock dataset. The network is converted for hardware and resource tests are performed for the different types of hardware.

The research conducts an in-depth sensitivity analysis to gauge the impact of various design choices on the rover's limited energy resources. Both centralized and decentralized swarm configurations were considered, alongside monocular and stereo vision systems. The operational modes of these systems were also rigorously tested. Under current operational assumptions, the OBCA design incorporating the XE2 emerges as the only viable solution that satisfies all requirements. However, this option incurs additional mass and cost implications. Thus, a centralized swarm approach is proposed, wherein a single rover performs computationally intensive tasks for multiple units. This approach necessitates significant upgrades to the communication subsystem and poses challenges in terms of reduced sys-

tem reliability due to centralization.

Future work should focus on developing and validating lightweight DL-based algorithms for Lunar Zebro and similar rovers, with testing in a simulated lunar environment. This includes creating a lunar testbed for validation of the rover's navigation and hazard detection systems, testing algorithmic performance under radiation exposure, and exploring efficient algorithmic implementations suitable for low-SWaP constraints.

Keywords: OBCA, DL, Hazard Detection, low-SWaP Space Missions, Lunar Zebro OBCA, Swarming, AI-based FDIR, Autonomous Mission Planning

Contents

List of Figures

List of Tables

Acronyms

AI	Artificial Intelligence	v, vii, 1, 2, 4, 9, 14, 23, 31, 41, 43, 45, 46, 50, 52, 73, 78, 89, 92–94
ANTS	Autonomous Nano-Technology Swarm	27
APF	Artificial Potential Field	10, 13, 50, 54, 55, 88, 90
API	Application Programming Interfaces	7
ASAS	Autonomous Soil Assessment System	25
ASIC	Application Specific Integrated Circuit	22, 41
AU	Astronomical Unit	56
B-AO	Bacteria-Aritificial Obstacle	88
BIN	Binary	68
BMS	Battery Management System	6, 46, 47
CADRE	Cooperative Autonomous Distributed Robotic Exploration	27
CAN	Controller Area Network	vii, 47–49, 92
CDH	Command and Data Handling	vii, 7, 38
CLI	Command Line Interface	68
CME	Coronal Mass Ejection	17
CMOS	Complementary Metal-Oxide-Semiconductor	20
CNN	Convolutional Neural Network	iii, vi, vii, 2, 12–15, 21, 22, 41–43, 52, 58, 61, 67, 84, 85, 94, 97
CNSA	Chinese National Space Administration	24
COTS	Cost of the Shelf	v, 1, 2, 4–6, 17, 23–25, 28–30, 33, 94
CPU	Central Processing Unit	iii, vii, ix, 8, 14, 21, 22, 25, 41, 52, 56, 58, 67, 80, 81, 85, 86, 96
DL	Deep Learning	1–4, 10, 29, 31, 43, 61, 63, 64, 82–84, 90, 92–94, 96, 97
DMIPS	Dhrystone million instructions per second	8
ECC	Error Correcting Code	7, 44

ECSS	European Cooperation for Space Standardization	32
EDAC	Error Detection and Correction	8, 35, 36, 85, 97
EPS	Electrical Power System	6, 31, 39, 42, 92
FDIR	Fault Detection, Isolation & Recovery	vi, x, 1–3, 20, 29, 31, 38, 44, 50–52, 77, 78, 82, 89, 90, 92–94
FP16	Floating-Point 16	66–68, 77, 88
FP32	Floating-Point 32	51, 87
FPGA	Field Programmable Gate Array	iii, vii, 7, 14, 21, 22, 25, 41–43, 67, 96
GCR	Galactic Cosmic Ray	ix, 17, 56, 59, 117
GPU	Graphics Processing Unit	vii, 21, 25, 41, 67
HDF5	Hierarchical Data Format version 5	67
IC	Integrated Circuit	48, 56
IMU	Inertial Measurement Unit	55
INCOSE	International Council on Systems Engineering	28
IoU	Intersection over Union	62, 65
IPC	Inter Process Communication	7
ISO	International Organization for Standardization	59
ISRO	Indian Space Research Organisation	1
JMP	Jumper	48
JPEG	Joint Photographic Experts Group	68
JPL	Jet Propulsion Laboratory	24
KLRD	Kernel Low-Rank Representation (KLRR)-based Rock Detection	13
KPRD	Kernel Principal Component Analysis (KPCA)-based Rock Detection	13
LCT	Lunar Cold Trap	19
LEO	Low Earth Orbit	1, 17, 23, 59
LMS	Locomotion System	6, 35, 47, 85
LPDDR	Low-Power Double Data Rate	44
LSTM	Long Short-Term Memory	50, 51
MCP	Master Control Program	7, 8, 85, 97
MCU	Microcontroller Unit	41, 47, 70
MIMD	Multiple Instruction, Multiple Data	26
NASA	National Aeronautics and Space Administration	1, 10, 13, 23–25, 27, 43
NCS	Neural Compute Stick	viii, 69
NPU	Neural Processing Unit	25

OBC	On-Board Computer	vi–xi, 2, 6–10, 20, 24–26, 30, 31, 33–35, 38–41, 43, 45–47, 49–51, 53–55, 58, 61, 69–71, 73–81, 84, 86, 90–92, 97
OBCA	On-Board Computing Architecture	iii, iv, vii, viii, x, 2–4, 6, 9–11, 16, 17, 20, 24, 28–33, 36, 37, 39, 40, 45, 47, 52, 53, 55, 56, 58, 59, 73, 79, 80, 83–85, 90–97
ONNX	Open Neural Network Exchange	67, 68
OS	Operating System	31, 86
PNG	Portable Network Graphics	68
PPU	Power Processing Unit	6–8, 34–36, 46–48, 85, 90
PS	Processing System	viii, 7, 25, 43, 55, 57, 58, 69, 74, 90, 93, 97
PSYCHIC	Prediction of Solar particle Yields for CHaracterizing Integrated Circuits	59
RAM	Random Access Memory	vi, 7, 21, 24, 33, 44, 73, 76, 83, 85–87
RAMS	Reliability, Availability, Maintainability, and Safety	32
RGB	Red, Green and Blue	68, 88, 90
RKLRR	Robust Kernel Low-Rank Representation	13
RNN	Recurrent Neural Network	51
ROCKSTER	Onboard Rock Segmentation Through Edge Regrouping	13
ROM	Read-Only Memory	21
RP2	Raspberry Pi 2	vii–x, 52, 53, 69, 70, 73–77, 84–86, 93, 114, 115
RS485	two wire, half duplex, multi-master communication protocol	vii, 7, 8, 31, 34, 46, 48, 92
SBC	Single-Board Computer	iii, vii, 21, 41–43, 46–48, 92, 96
SEE	Single Event Effects	20, 34
SEFI	Single-event Functional Interrupt	20
SEL	Single-event Latch-up	20, 59
SET	Single-event Transient	20
SEU	Single-event Upset	11, 20, 47, 56–59, 93, 97

SHAVE	Streaming Hybrid Architecture Vector Engine core	22
SHRIMP	Small High-Resolution Independent Modular Photographer	8, 10, 21, 36, 46, 47, 65, 77, 83, 84, 88, 93
SMART	Specific, Measurable, Attainable, Realistic and Traceable	31
SoC	System on Chip	vii, x, 7, 9, 21, 22, 24, 25, 41, 43, 57, 58, 69
SOTA	State Of The Art	13, 24, 42, 51, 56, 67
SPE	Solar Particle Event	17
SPENVIS	SPace ENVironment Information System	56, 57, 59
SWaP	low Size, Weight, and Power	iii, 1, 2, 4, 13, 21, 24, 29, 39, 41, 84, 92, 93, 96
TCN	Temporal Convolutional Network	51, 52, 58, 90
TID	Total Ionizing Dose	20, 34, 59, 93
TPU	Tensor Processing Unit	vii, 21–23, 41–43
UNN	Ubotica Neural Network	67, 68
VPU	Vision Processing Unit	iii, vii, 21–25, 41–43, 46, 47, 51, 67, 92
YOLO	You Only Look Once	36

1 Introduction

The Moon, historically a symbol of human aspiration and curiosity, has consistently held a central role in our quest for space exploration. In recent years, there has been a notable resurgence in lunar exploration initiatives, with ambitions from governmental organisations like National Aeronautics and Space Administration (NASA) [13] and Indian Space Research Organisation (ISRO) [13]. This renewed interest is not only driven by national space agencies but also by the burgeoning private sector [14] and academic institutions [15, 16].

One such innovative project emerging from this renewed lunar exploration era is the Lunar Zebro mission, spearheaded by a dedicated team of students from TU Delft [16]. The Lunar Zebro project distinguishes itself with its bold vision of deploying a swarm of miniaturized, low Size, Weight, and Power (SWaP) rovers on the lunar surface [17]. These rovers, emblematic of the technological advancements in the field, aim to revolutionize lunar exploration by providing a versatile, efficient, and scalable approach to lunar surface investigation.

In the broader scope of the Lunar Zebro project, the team envisions the potential integration of advanced Deep Learning (DL) techniques as a supplementary enhancement to the rover's operational framework, aiming to explore the frontier of Artificial Intelligence (AI) applications in lunar exploration. The incorporation of DL, particularly in hazard detection, presents an opportunity to augment the rover's autonomy and adaptability in the challenging lunar environment.

The feasibility of this ambitious endeavor is significantly bolstered by the ongoing NewSpace revolution [18], which emphasizes the utilization of Cost of the Shelf (COTS) products. This paradigm shift in space technology development has democratized access to advanced computational hardware, making it viable to incorporate sophisticated AI algorithms into compact and cost-effective lunar rovers. Ubotica Technologies, a company specializing in the utilization of COTS AI accelerators for Low Earth Orbit (LEO) missions, is keenly focused on exploring their application in lunar expeditions, such as the Lunar Zebro mission. The execution of this thesis is supported by Ubotica Technologies, reflecting a collaborative effort to advance the understanding and implementation of these innovative technologies in lunar exploration.

In pursuit of enhancing the Lunar Zebro mission, a comprehensive literature study was conducted to explore the integration and impact of AI and deep learning across various facets of the rover's functionality. This study aimed to establish a foundational understanding of AI in space exploration, particularly for lunar missions. It encompassed research into DL-based Fault Detection, Isolation & Recovery (FDIR) systems for subsystem health monitoring, the potential of AI in managing swarm behavior, and autonomous path planning. Moreover, the use of COTS hardware in space is researched. Since the implementation of DL algorithms for hazard detection on such a constrained platform as Lunar Zebro has not been researched before, this research is the main focus of this MSc thesis.

Therefore, this project sets out to explore the implementation of deep learning algorithms in the context of the Lunar Zebro mission. It aims to assess the potential enhancements in the rover's capabilities and evaluate the technical and practical aspects of running advanced AI models onboard a lunar rover. Through this exploration, the thesis seeks to contribute to the broader narrative of lunar exploration, showcasing the integration of cutting-edge AI technologies in space missions with a special focus on the constrained resources of Lunar Zebro.

Having introduced the general domain and research topic of this thesis, a review of relevant literature is presented in chapter 2, with a special focus on the research gap that serves as the starting point for this research. Chapter 3 follows the systems engineering approach and describes the conception of system requirements for the On-Board Computing Architecture (OBCA) of the lunar nano-rover. This chapter also discusses various design options considered. The detailed design of the OBCA, including the design of the data bus, preliminary software design, and a radiation risk assessment, is discussed in chapter 4. In chapter 5, the training and implementation of a rock detection Convolutional Neural Network (CNN) on different hardware platforms are discussed. Additionally, this chapter outlines the test method for measuring the required resources of the On-Board Computer (OBC). Finally, chapter 6 discusses the results of these tests. These results are contextualized by running simulations of the rover's operations, assessing whether the rover can function within the set operational limits. Different swarm configurations and operational modes of the AI accelerator are considered in this analysis. Moreover, this final chapter includes system verification based on the established requirements.

1.1. Research Outline

With the advancements and challenges outlined, the research objective of this thesis is:

To contribute to the body of knowledge about the implementation of DL algorithms on low-SWaP systems (<2 kg) in deep space.

As will be elaborately outlined in chapter 2 (section 2.2 & section 2.4 specifically), the main knowledge gaps that were identified in the context of a low-SWaP (<2 kg) rover in deep space:

1. The feasibility of DL-based hazard detection & FDIR on housekeeping data.
2. Using COTS AI-accelerators.

Based on the findings from the background research in chapter 2 described in this report, the research question was formulated. The supporting sub-questions aim to provide further clarification and insight into the most crucial aspects of the investigation, thereby directing the course of the study.

How can the OBCA for the next-generation Lunar Zebro be designed cost-effectively?

Where next-generation refers to the capability of the rover to function in a swarm and to apply DL-based hazard detection and FDIR. Moreover, the cost-effectiveness of the project is of crucial importance to make the low-budget Lunar Zebro project a reality. It should be emphasized that while this study primarily concentrates on the OBCA design for the Lunar Zebro as a specific case, the broader application of this subsystem's design to extraterrestrial nano-rovers in general is also of interest, as this addresses a more general, overarching question relevant to the field of extraterrestrial rover design.

This main objective can be complemented by the following sub-questions:

1. *What are the required resources for DL-based hazard detection, and how can the OBCA design under investigation provide these resources (respecting the SWaP constraints)?*

   As discussed in section 2.2, there are many ways to recognize hazards. This subquestion refers to the requirements posed on the OBCA by a relevant hazard detection algorithm.

2. *How can the OBCA facilitate DL-based FDIR, swarming, path planning and autonomous mission planning?*

   A crucial aspect of this thesis involves ensuring that the Lunar Zebro is equipped with a computational system that can facilitate its autonomous functions. The rover should be able to find its way across the Moon's landscape without commands from Earth, and therefore swarming, path planning, and autonomous mission planning operations should be carefully taken into account when designing the OBCA. Moreover, the reliability of the rover could be much increased by advanced FDIR on subsystem housekeeping data.

3. *Should the OBCA be designed homogeneously across the whole swarm?*

   In section 2.5, it will be described that the swarm can function as a decentralized whole, or it can have a central computational hub with different computational capabilities. The choice for the swarm configuration will have consequences for the design of the OBCA. This is elaborately discussed in section 2.5 and subsection 4.3.7.

2
Background

This chapter is divided into 6 sections. The Lunar Zebro mission concept is discussed in section 2.1, as the existing mission will shape the requirements for the OBCA. The system architecture and design features are outlined in subsection 2.1.2 and the mission phases in subsection 2.1.3. In section 2.2 hazard detection methods are outlined, and it is highlighted that the a study of the feasibility of running DL-based algorithms on a low-SWaP mission as Lunar Zebro is a research gap. Section 2.3 discusses the challenges of the lunar environment for the OBCA design, followed by an overview of radiation effects and fault-tolerant design techniques in section 2.4. In this section, the types of hardware possibilities for are discussed, with a special focus on AI-accelerating and COTS hardware. Because the next generation of Lunar Zebro will function as part of a swarm, the swarming concept and practical examples are discussed in section 2.5.

2.1. Lunar Zebro

Lunar Zebro, initiated by TU Delft, is a lunar nano-rover (<1.5 kg [19]) that is meant to launch soon after the start of 2025 [16]. While for the first mission, one rover will perform a technology demonstration, in the years thereafter the Lunar Zebro team is planning on launching a second generation of Lunar Zebro to the Moon, in the form of a swarm [17].

2.1.1. The Mission and Capabilities

The first mission of Lunar Zebro aims to deploy the rover on the moon for a full lunar day, which is about 14 Earth days. This mission [20] is a scientific venture as well as a technology demonstration, intended to carry out radiation measurements and test a new locomotion system specifically designed for the challenging lunar terrain [21]. The rover will also be showcasing the effectiveness of an exploratory vehicle under SWaP constraints.

Looking beyond its initial mission, Lunar Zebro is set to evolve into an exploratory project involving swarm-based lunar exploration. This approach envisages deploying multiple rovers to explore the lunar surface, either independently or in careful conjunction with human missions [22], which would increase the chance of mission success as the human workload is kept to a manageable level [23]. As part of the technology demonstration of this mission, the swarm of rovers will designed to function autonomously, meaning that no human interaction or direct contact with Earth is necessary for mission success. This demand is reflected in requirements **LZ-OBCA-SH-013** and **LZ-OBCA-SYS-002** in Table 3.2 and subsection 3.1.3.

The ambition to keep the rover as small and light as possible stems from the enormous costs that come with the crossing to the Moon. Astrobotic, a private company that provides services for delivering payloads to the Moon, offers to bring a kilogram to the Moon for $1,200,000 [14]. Since the chassis of the rover mainly consists of Aluminum ($\rho = 2.7 g/cm^3$), every cubic centimeter of chassis that needs to be brought to the Moon would cost about $3600 extra, underlining the importance of keeping the rover

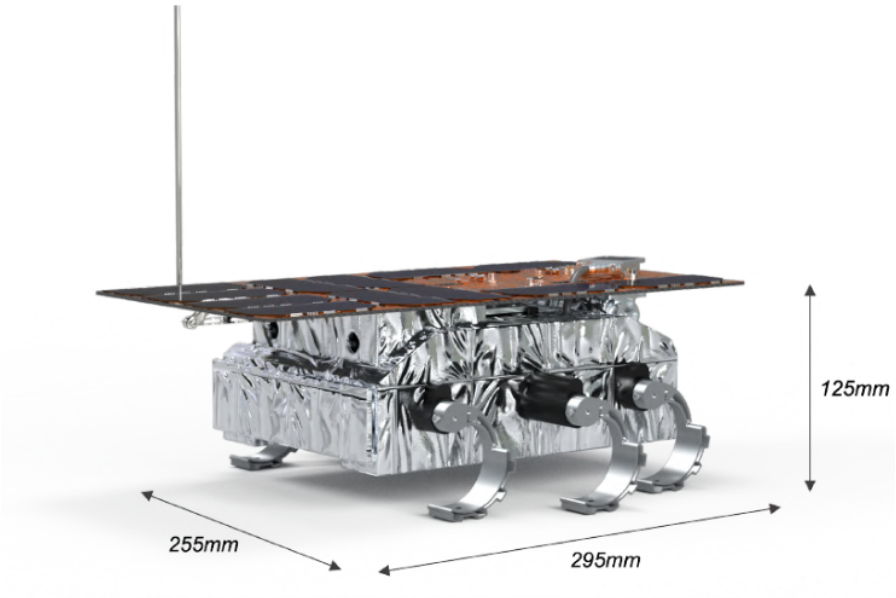


Figure 2.1: The Strelka rover

small, especially for a student team with a limited budget.

Keeping the mass of the rover low is a priority for the Lunar Zebro team. Especially because small increases in mass of one rover, would multiply by the number of rovers in the swarm. Even more, small increases in the mass of space missions accumulate rapidly due to the structural and propellant mass increases that are required to carry the vehicle into space and support the mass of the subsystems during operation. Thompson et al. demonstrated that every additional kilogram on the lunar ascent module of the Saturn V launch vehicle resulted in an over 800-fold increase in the total mass [24], stressing the importance of designing for a low mass of the swarm. This compounding of mass on space vehicles is also known as the 'Snowball Effect'.

2.1.2. System Architecture and Main Design Features

Lunar Zebro's design is characterized by its small, lightweight, and modular structure, integrating both COTS components and in-house developed modules. This approach aligns with the NewSpace philosophy, focusing on cost-effectiveness and rapid development [25]. The design of the second generation will build upon the design of the first rover, which is not set in stone yet, which asks for careful consideration of possible changes in the current design [26], as these will influence the design of other subsystems.

The rover consists of seven main subsystems, which are depicted in Figure 2.2.

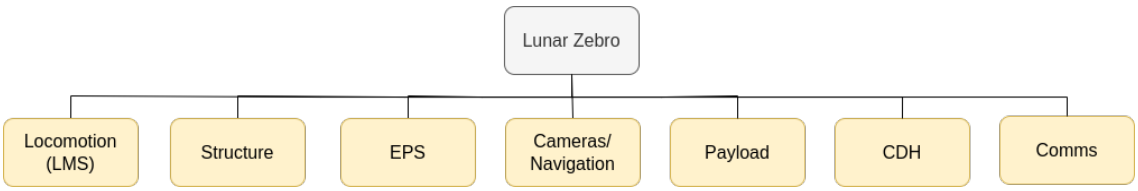


Figure 2.2: Subsystems of Lunar Zebro [26]

Locomotion System

A key feature is its specialized locomotion system, with six C-shaped legs that can rotate in both directions. The locomotion system was optimized for reliability and robustness [21], meaning that it will

not sink into the lunar regolith and safely overcome obstacles up to 3 cm in height [26], such as in Figure 2.3 [21]. The system was also designed to be light-weight ($89\,\mathrm{g}$ for the total system) while consisting of only COTS parts [21]. The system consumes a total of $23\,\mathrm{W}$ during operation, at $12\,\mathrm{V}$. The Locomotion System (LMS) can reach up to 5 cm/s, but the average velocity of the rover is expected to be 2 cm/s.

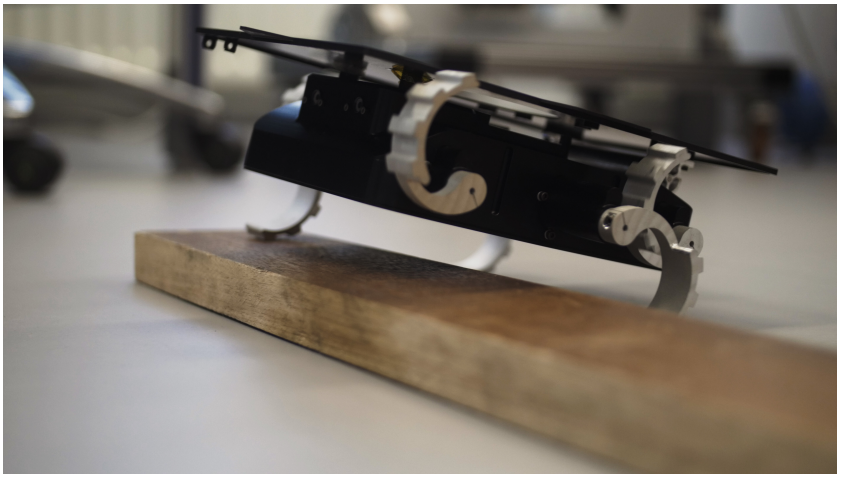


Figure 2.3: Engineering Model Belka-1 crossing an obstacle [21]

Electrical Power System (EPS)

The EPS of Lunar Zebro consists of the Battery Management System (BMS), the battery pack, the solar array and the Power Processing Unit (PPU) [26].

Other than stated in the design documentation of Lunar Zebro [26], M. Hubers (personal communication, December 2023), who is responsible for the design of the EPS of Lunar Zebro, indicated the newest design of the rover houses five instead of four NCR18650B Panasonic Lithium-Ion batteries [27], arranged in series.

Peak power refers to the highest level of charge and discharge energy that the battery can sustain momentarily without surpassing predetermined battery constraints. While the peak power of one cell on board Lunar Zebro is equal to $36\,\mathrm{W}$ [27], a series connection of the cells allows for a peak power of $P_{peak_{circuit}} = P_{peak_{cell}} \times N_{cell}$. The peak power of the whole battery pack is therefore $180\,\mathrm{W}$. It should be noted that a series connection of the batteries is not standard. Many other missions have the batteries connected in parallel [28, 29]. This imposes higher requirements on the peak power draw of the OBCA and shall be taken into consideration in the design of the OBCA.

The subsystem's maximum power draw is governed by the battery system's peak power and the converters' maximum output capacity. The power is distributed to the subsystems of the rover through three converters according to M. Hubers (personal communication, December 2, 2023), each with an own supply voltage; $3.3\,\mathrm{V}$, $5\,\mathrm{V}$, $12\,\mathrm{V}$. These converters can deliver up to $14.4\,\mathrm{W}$, $20\,\mathrm{W}$ and $48\,\mathrm{W}$, respectively.

The microcontroller that controls the rover's power distribution is called the PPU. It manages the power switches for every subsystem and keeps an eye on the voltage and current levels. To obtain power-related housekeeping data, the OBC connects with the PPU via the PPU app. If necessary, the central OBC can also ask the PPU to turn on or off a subsystem. When everything is operating normally, the PPU is also in charge of coordinating with the BMS), through which battery information is transferred from the PPU to the main OBC software [26].

Command and Data Handling (CDH) System

The rover will be equipped with the All-Programmable System-on-Chip featuring the Xilinx Zynq-7020, the Xiphos Q7S [30], which is depicted in Figure 2.4 [31]. The Field Programmable Gate Array (FPGA) houses a dual-core ARM Cortex-A9 processor, also referred to as the Processing System (PS). The System on Chip (SoC) has 1x512 MB (256 MB with Error Correcting Code (ECC)) and 1x256 MB low-power double-data-rate memory Random Access Memory (RAM) chips, 2 MicroSD slots (max. 32 GB each) on independent buses/power control, 2x128 MB QSPI Flash, and an external mass memory interface [30].

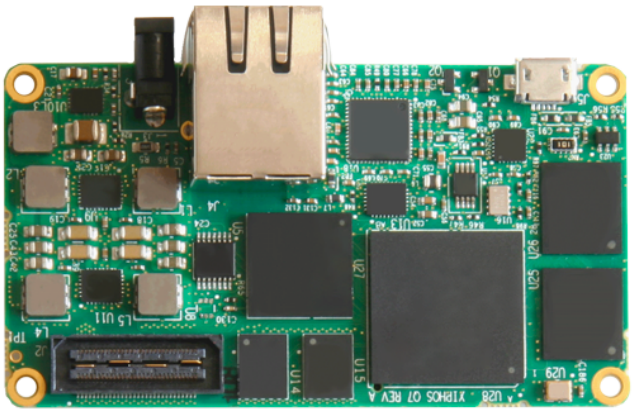


Figure 2.4: The OBC for the first generation of Lunar Zebro: the Xilinx Xiphos Q7S [31]

The Lunar Zebro team chose for one OBC for the sake of simple architecture, rather than choosing for a redundant OBC [26]. However, the PPU has been identified as a viable alternative to assume the responsibilities of the OBC in scenarios where the OBC is not operational, albeit with certain limitations in its capacity.

Nominal Control Software Architecture

During normal operations, the OBC operates as the primary controller within the CDH subsystem. The core component is supported by individual applications tailored for each submodule. The core system comprises two key elements: the Master Control Program (MCP) and a Router, and the comprehensive architecture is illustrated in Figure 2.5 [26].

The MCP encompasses the Process Manager, Application Programming Interfaces (API) for the submodules, and the State Machine. The Process Manager oversees all active processes running on the OBC. The APIs serve as standardized data formats for facilitating communication with the submodules. Meanwhile, the Router plays a pivotal role in transmitting and receiving messages between TRON and the submodules. The OBC software also provides an Inter Process Communication (IPC) library to the submodules, simplifying the implementation of communication protocols through the Router [26].

The individual applications establish communication with their respective hardware components through two wire, half duplex, multi-master communication protocol (RS485) buses, as depicted in the lower layers of the architecture. Regarding the State Machine, it defines the decision-making processes for executing operational sequences. Multiple state machines have been developed to cater to various tasks that the Zebro is expected to perform on the Moon [26]. Since all subsystems are developed with RS485 protocol for communication lines, this protocol will be assumed for the next-generation rover as

well. This poses requirements on the communication within the rover, the creation of the data bus, and the interfacing of the central OBC. This demand is reflected in the interface requirements **LZ-OBCA-INT-001** to **LZ-OBCA-INT-004** in subsection 3.1.3. RS485 is a serial communication protocol that in a half-duplex mode, allows bidirectional but not simultaneous data transmission over a single pair of wires [32].

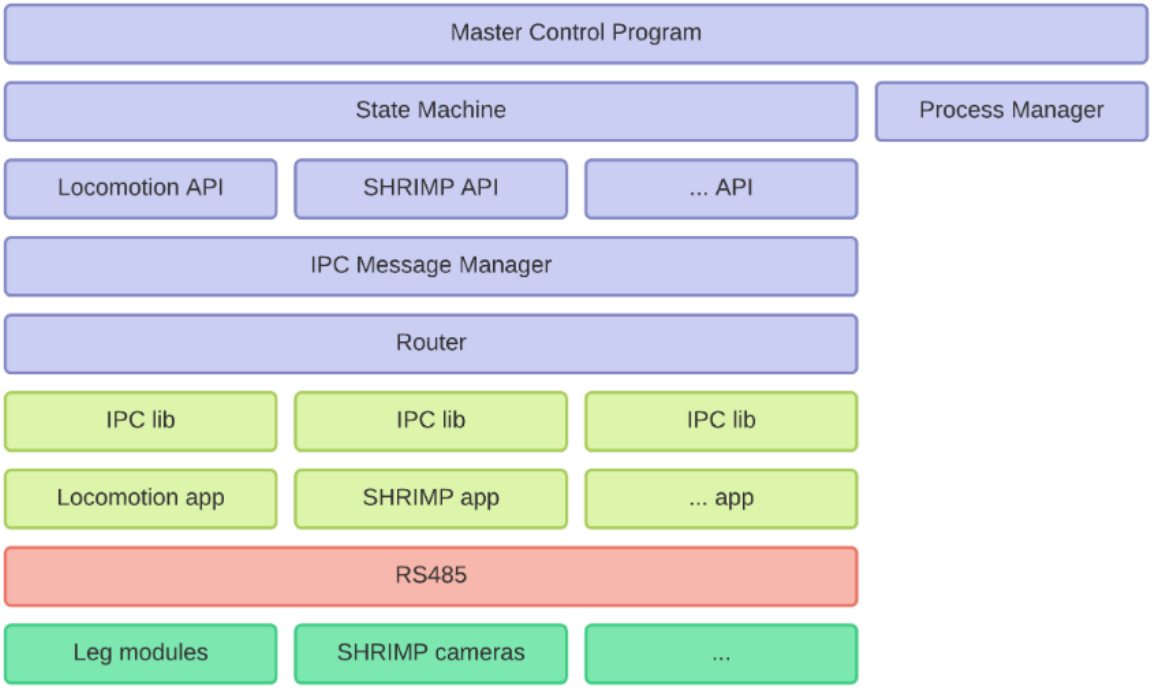


Figure 2.5: The central OBC software architecture as implemented on the first mission [26]

One of the major design decisions for the first software implementation on-board Lunar Zebro, was to create a modular design of applications around the MCP [26]. This decision was made due to the nature of the project, in which students are responsible for separate parts of the rover. The OBC system architecture, with all modular applications, is depicted in Figure 2.6.

Estimations by the Lunar Zebro team (personal communication, Y. KLaassen, December 2023) indicated that 5% of the Zynq 7020 Central Processing Unit (CPU) core load is sufficient to continuously control the motor drivers of the rover. Since the ARM Cortex A9 dual-core of the Xiphos Q7S houses 4150 Dhrystone million instructions per second (DMIPS) [33], this indicates that 207.5 DMIPS are required for control of the locomotion motor drivers. Besides the continuous control of the motor drivers, several other apps require continuous processing power: the Master Control Program, the Error Detection and Correction (EDAC) for the on-board computing architecture and the PPU application. Knowing that the locomotion app requires 5%, it is assumed that these apps require 10%, 5% and 5% continuously. Since the full-time, continuous use of these applications is an overestimation, this is assumed to be a safe margin for operation. Therefore, a requirement shall be set that states that 0.25×4150 DMIPS $= 1038$ DMIPS are required for operation of applications other than communications, hazard detection, the Small High-Resolution Independent Modular Photographer (SHRIMP) and payload apps and path planning. This demand is reflected in requirement **LZ-OBCA-OPER-003** in Table 6.12. The communications, hazard detection, the SHRIMP, payload, and path planning applications will be part of the standard static operational cycle of the rover, which will be discussed in subsection 3.1.5.

Payload

The first generation of Lunar Zebro will have a miniaturized radiation-measuring payload on board [26, 34]. This specially designed payload would acquire $0.8\,\mathrm{MB}$ of data per 24 hours and $11.2\,\mathrm{MB}$ in 14

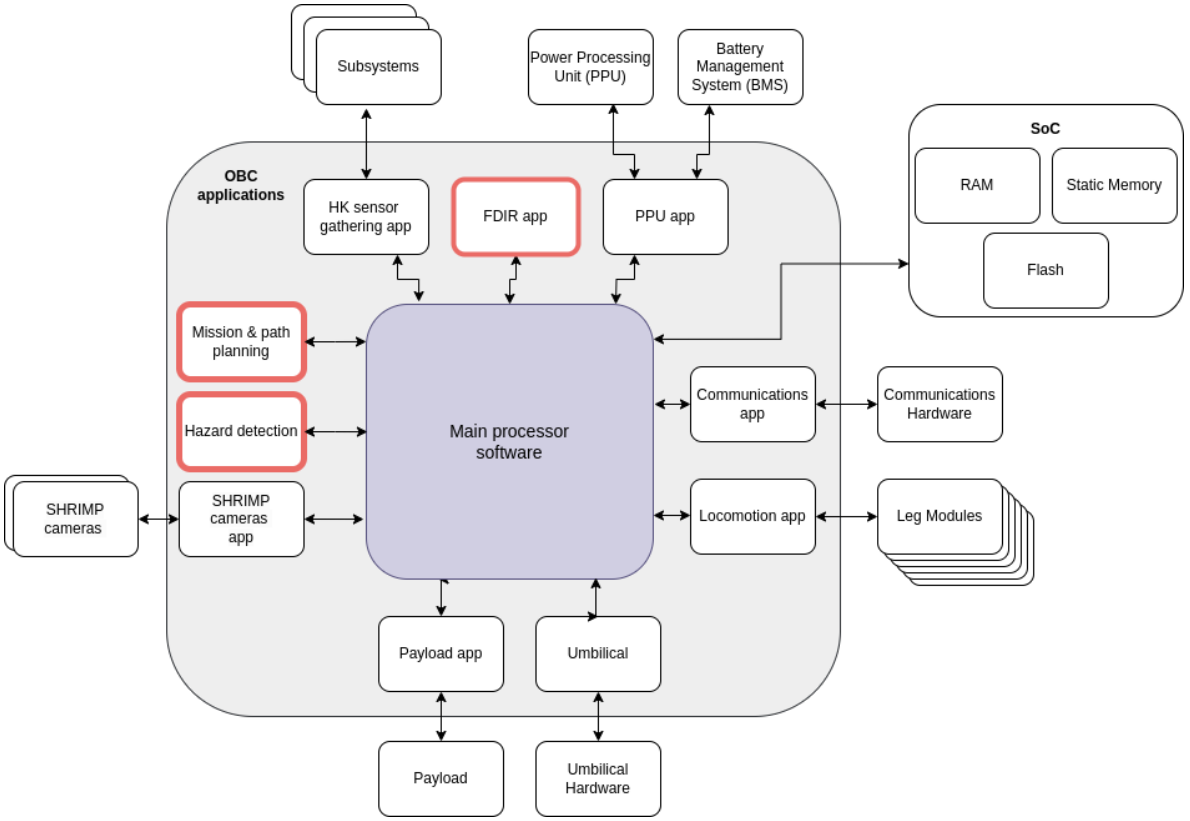


Figure 2.6: The OBC system architecture. The blocks within the light-shaded area represent applications that run on the OBC. The SoC-block represents the rest of the relevant hardware that will definitely be found on the SoC. The red boxes represent applications that could possibly (partly) be performed on an AI-accelerator. Inspired by [26].

days, with a sampling rate of 1 min^{-1} [34]. This is a conservative estimate, taking into account overhead for any inter-subsystem communication protocols. Moreover, the data sampling rate is taken to be much higher than necessary on the mission here, to estimate the total gathered data conservatively.

For the next generation of Lunar Zebro, a radiation payload will probably not suffice. This rover will have exploratory purposes, likely supporting in pinpointing locations of certain resources on the moon. To thus be able to support such a payload with the future OBCA, payloads with the potential for higher data collection should be taken into account. An example of an interesting payload for this resource analysis is a mass spectrometer. Research has been done into the miniaturization of such a payload for planetary rover applications [35]. Rohner et al. [35] developed a laser ablation payload with a mass of approximately $280\,\mathrm{g}$ and operating at an estimated power of only $3\,\mathrm{W}$. At its maximum time resolution of $2.5\,\mathrm{ns}$, the payload requires a storage space of $128\,\mathrm{kB}$ per measurement. Assuming 1 measurement per minute during operational periods in between solar charging periods, the total required storage space would be $702\,\mathrm{MB}$. This is a very conservative estimate again, as the measurement frequency would be much lower; at the low average velocity of the rover (2 cm/s), the rover would need to cover more distance before taking another measurement. This payload would only be activated when the rover reaches its scientific goal, which is most likely only reached after covering larger distances. Moreover, this payload would require the rover to be stationary during measurements, decreasing the average velocity of the rover even further if too many measurements are taken.

For the OBCA on board the next-generation mission of Lunar Zebro, it is important to consider future implementations and especially the worst-case scenarios amongst these possible future implementations. Hence, the design of onboard data connectivity and storage accounts for the mass spectrometer data rate requirements. It is assumed that there will be no more than one payload per rover, as this would increase the mass and cost per rover quickly, as discussed in subsection 2.1.1.

Communications Subsystem

The Communications System for the first rover consisted of a radiation communication system for direct communication with small ground stations on Earth [26], which was developed for lunar microsatellites Longjiang-1 and Longjiang-2 [36]. However, the second generation functions completely autonomously, meaning that communication between the rovers is required, but communication to Earth is not necessary. Therefore, a short-range radio system like the bi-directional ZigBee communication system that was also implemented on the Ingenuity helicopter and Perseverance Rover [37] of NASA will be a better fit for this mission. This communications system allows for data rates up to 250 kbps and allows for omnidirectional reception and transmission. This omni-directionality is required for the functioning of the swarm, in which individuals will share their locations through the communication system.

Navigation Subsystem

The Navigation subsystem consists of two small cameras SHRIMP, and corresponding object detection software. This software is meant to recognize rocks that are larger than 3 cm in height and consists of a classic object detection method that is described in more detail by James in [26]. Although this classic method was worked out in detail, it was never tested A. Ayata stated (personal communication, December 2023) on image data representative for the SHRIMP cameras, therefore details on the performance of this algorithm cannot be given. As part of the science demonstration for the next generation of Lunar Zebro, a requirement is posed stating that the hazard detection algorithm should be DL-based. DL-based hazard detection methods and a comparison with classic algorithms are described in the next section, section 2.2.

The path planning algorithms were developed in the works of Geeling [38] and Manteaux [39]. In their work, an Artificial Potential Field (APF) algorithm was chosen. This algorithm requires the rover to register the exact size of obstacles, as only then the appropriate repulsion force can be pinpointed at the obstacle location in the map. This poses an important requirement on the hazard detection algorithm; the exact location and size of the obstacles should be determined. This demand is reflected in requirement **LZ-OBCA-FUN-010** in Table 6.12.

2.1.3. Mission Phases

Next, the phases of the mission should be outlined to understand exactly what mission profile the OBCA should be designed for. These details are outlined in Table 2.1, which offers a concise overview of each mission phase. Within the table, each phase is briefly described, highlighting its significance in relation to the mission, specifically concerning the OBCA. Additionally, the table denotes the transition events between phases and specifies the expected duration of each phase. A typical mission profile for a lunar mission is depicted in Figure 2.7 [40]. For the initial 5 stages, the rover is protected by a launch vehicle and/or lunar lander like the Peregrine lander designed by Astrobotic [14].

Table 2.1: The different phases of a lunar mission with specifications.

Phase	Description	Duration	OBC State
Pre-Launch Phase	Preparation for launch.	N/A	OFF
Launch	Injection from Earth surface into Earth orbit.	1-3 hours	OFF
Earth Orbit	Stable orbit around Earth. Time for checks.	4 days [14]	
Cis-Lunar Transit	Transfer from Earth to the Moon.	5 days to 2 months. [40]	OFF
Lunar Orbit	Stable orbit around the Moon.	6 days to 2 months.	OFF
Lunar Surface	Operations on the Lunar Surface. [40]	ON	

The following six mission phases can be recognized:

- **Pre-Launch Phase:** During the pre-launch phase, the rover is integrated into the spacecraft and undergoes various tests to ensure its functionality in the lunar environment.
- **Launch Phase:** For the launch phase, the mission requirements are dependent on the choice of launch vehicle. The OBCA needs to withstand high levels of vibration and mechanical stress

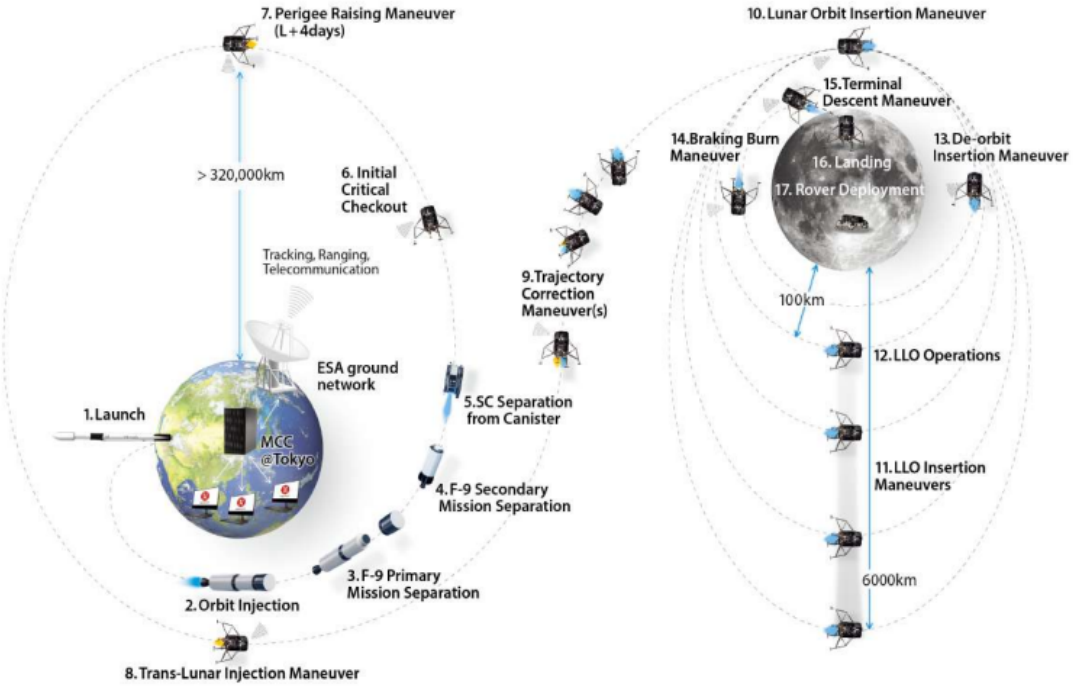


Figure 2.7: Exemplary mission profile for a lunar lander [40]

during the launch. Design considerations for the OBCA involve ensuring its resilience to these vibrations. The OBCA will be inactive during this period, as well as during following periods, until safe landing on the lunar surface.

- **Earth Orbit:** After launch, the lander with the rovers on board spends time in Earth's orbit, conducting checks on all payloads that it hosts. Possible first radiation measurements done by the radiation payload will be done passively and these can be carried on until end-of-life. The phase ends with a trans-lunar injection burn. The near-Earth radiation environment is dominated by trapped radiation in the van Allen belts. The received radiation is expected to be about $20\,\mathrm{rad\,d^{-1}}$.
- **Cis-lunar Transit:** Due to a trans-lunar injection maneuver, the spacecraft is sent on its way to the Moon. This transit can take up to 30 days [14].
- **Lunar Orbit:** The lunar orbit phase can take up to 25 days [14] and encompasses all actions related to placing a payload into lunar orbit, deploying orbital payloads, and descending to the surface of the moon.
- **Lunar Surface:** Only just before or even after deployment of the rover on the surface, the OBCA will be activated. The lunar surface phase entails surface operations like deploying rovers, acquiring payload data, and capturing images of the lunar surface.

It is essential to identify the specific mission phases during which electronic devices are activated, as their susceptibility to radiation-induced damage is significantly higher when operational. In the powered-on state, the circuits are more vulnerable to transient effects like Single-event Upsets (SEUs), where the passage of a charged particle can cause temporary malfunctions or even permanent damage. Therefore, knowing the operational timeline of these devices enables engineers to devise strategies to minimize their active time during high-radiation segments of the mission, thereby reducing the risk of radiation-induced failures.

2.2. Hazard Detection

For Lunar Zebro, hazards on the Moon's surface entail rocks, craters, or holes and slopes. In this project, the detection of rocks will be taken as a starting point. Following the demand that the swarm of Lunar Zebros is capable of operation without human intervention nor direct control with any ground station, the autonomous capacity to detect and categorize lunar rocks is fundamental for the Lunar Zebro mission's success. The taxonomy of computer vision can be categorized into three main tasks: classification, object detection, and segmentation [41], as depicted in Figure 2.8 [42].

- **Classification** is a process that involves assigning specific labels or categories to objects based on their defining characteristics. The term classification is also used often for segmentation tasks that 'classify' images into different objects or types, such as by Cross et al. [3], however in this work, this will be referred to as segmentation, as it is a pixel-by-pixel operation.
- **Object Detection** employs models to identify and locate objects within an image, typically by creating bounding boxes around them, each associated with a class label. However, these bounding boxes are limited to rectangular or square shapes, and they do not convey specific details about the object's shape. This approach offers a high-level categorization of objects but lacks fine-grained information about their geometrical attributes.
- **Segmentation** models produce pixel-wise masks for each object present in an image. This technique allows for a more detailed and granular understanding of object boundaries and shapes, as it accurately delineates each object's contours within the image. Image segmentation thus provides a higher level of detail about the objects, making it particularly valuable when shape and precise boundaries are of significant interest. The computational complexity of segmentation networks is usually higher than those for object detection, thus leading to higher precision.

Figure 2.8: Three different forms of computer vision for detection of objects [42]

These approaches can be further categorized into classic and learning-based algorithms [43, 44]. Deep learning methods, involving CNNs, have shown superior performance in image segmentation tasks [45, 46], including the segmentation of rocks [2]. They are particularly effective in addressing challenges such as faint edge detection and multispectral image registration [44]. However, they can be resource-intensive and may struggle to perform outside their training space. Classic methods, on the other hand, are more transparent and can be more easily adapted for different purposes, as large quantities of annotated data are not necessary [44]. Despite these advantages, they often suffer from poor accuracy

and robustness, especially in the presence of noise (i.e. sensor imperfections, environmental conditions, or transmission errors) [45, 47].

Although quite some studies have been performed into unsupervised- [48] and even deep-learning algorithms [49] for extraterrestrial landers and rovers, an actual study of the feasibility of running such algorithms on board such a SWaP-constrained rover as Lunar Zebro has not been performed yet, therefore leaving a gap in the current body of knowledge.

This research specifically focuses on image segmentation over object detection for the Lunar Zebro. The decision is driven by the need for precise and detailed understanding of rock shapes and sizes in the lunar environment, which is crucial for effective navigation and obstacle avoidance (APF path planning requires precise obstacle sizes), despite the higher computational demands of segmentation. Moreover, segmentation networks have been used to also recognize craters/slopes as depicted in Figure 2.9, which is in accordance with stakeholder requirement **LZ-OBCA-SH-010** for this mission. This choice also aligns with the project's aim to explore advanced and computationally heavy computer vision techniques within the constraints of a low-SWaP system, thereby contributing to the field by addressing a significant research gap.

2.2.1. Survey of Rock Detection Methods

Most of recent innovations into extraterrestrial hazard detection methods have come from missions aimed for Mars [50, 2]. As described in section 2.3, since the lunar terrain is comparable to that of Mars, these innovations can be taken as an inspiration and performance indicators can be said to be indicative.

Consistently, literature stresses the difficulty of extracting rock boundaries through edge extraction. Factors that complexify this include the diversity in for example morphology, intensity, and texture of the soil [51, 52, 53]. Other common problems include shadows, dust covering, and boundary-blurring [54, 55].

On the Opportunity rover of NASA that was deployed on Mars in 2004, the Onboard Rock Segmentation Through Edge Regrouping (ROCKSTER) algorithm was used for the perception of rocks. Burl et al [53] stated that "ROCKSTER is arguably the most sophisticated perception algorithm for scientific analysis that has ever been deployed to another planet", and the algorithm is TRL 9 "mission proven". However, the algorithm has its deficits; noise and blurred rock boundaries. Therefore, the complete perimeter of the rock cannot always be detected. ROCKSTER uses the local gradient between pixels to recognize rock boundaries. As the name states, this focuses on a local shift in pixel values. However, These "local gradient operators" do not use statistical features such as "spatial layout and the inner relationship of pixels or regions" [53]. Spatial layout refers to the arrangement or distribution of pixels in an image. The inner relationship of pixels refers to how pixels are related to each other in terms of their spatial position, color, and intensity.

In light of the foregoing discussion, Xiao et al. [56] proposed to solve the rock detection problem by using several low-level features. Two novel rock detection methods were used: Kernel Principal Component Analysis (KPCA)-based Rock Detection (KPRD) and Kernel Low-Rank Representation (KLRR)-based Rock Detection (KLRD). KPRD is specialized in real-time detection but with less accurate results than KLRD, where KLRD has longer processing times. These two algorithms were compared with a State Of The Art (SOTA) kernel-based algorithm [57], and a SOTA deep-learning-based algorithm [58]. Both achieved superior performance compared to the SOTA based on the MarsData dataset [56], for which Mars rover images were collected. It should be noted that where the deep-learning-based method was developed to perform well on rock detection on Mars rover data, the Robust Kernel Low-Rank Representation (RKLRR) method was not necessary.

The introduction of CNN-based methods, especially U-Nets, marked a significant step forward in rock detection. U-Nets, originally successful in medical image segmentation, were well-suited for rock segmentation due to similarities in the unstructured semantic composition of the targets and the small-scale data volume. CNNs allow for segmentation, without the need for 'handcrafted feature extraction' [59].

Such convolutional networks have improved accuracy in rock segmentation 10-20% [2] at the cost of computational complexity. Notable variants like MultiResUNet [4] offered an improvement in feature integration. This model uses a series of convolution layers with residual connections to reduce the semantic gap in feature fusion, making it particularly effective for rock detection. MultiResUNet's architecture is relatively lightweight compared to a normal U-Net, primarily due to its efficient use of convolutional layers and reduced parameter count, which contributes to its suitability for deployment in resource-constrained environments like space missions.

Furthermore, the multiscale feature learning capability of MultiResUNet makes it an excellent candidate for detecting a variety of hazards beyond just rocks. Its design to capture features at multiple scales through different resolution paths enables it to effectively identify diverse features like craters, fissures, and other hazards in lunar or Martian terrains. This enhanced feature extraction capability, stemming from its elaborate architecture, ensures more comprehensive scene understanding, essential for navigation and hazard avoidance in extraterrestrial exploration. Thus, the adaptability of MultiResUNet to capture multiscale information positions makes it a versatile tool.

Advanced algorithms like RockFormer [2], integrating CNNs and Transformers, are making promising strides in extraterrestrial hazard detection. However, their sophisticated architecture, while effective in capturing detailed features, often faces compatibility challenges with the AI-accelerating hardware used in space missions. This poses a significant hurdle for their deployment in resource-constrained environments like lunar or Martian rovers, where computational efficiency and power limitations are critical. Research into lightweight networks for rock detection on the Moon is starting up, but networks with much less parameters tend to not recognize small-size rocks [60], which makes them unfeasible for Lunar Zebro, which cannot surpass obstacles larger than 3 cm.

A Canadian company named Mission Control studied the application of CNNs to classify terrain into several terrain types, and even did applied research on real-time systems [61]. Although classifying terrain types is not as relevant for Lunar Zebro because of its specially designed locomotion system, this classification shows how CNNs can use semantic information to classify different rock or terrain types. This semantic information can even be used to detect novel geologic features for scientific interest [3, 62]. Such geological categorisation networks have even been deployed onto a Xiphos Q8 computing hardware [62], a device very similar to the Xiphos Q7s, the device that Lunar Zebro will house during its first mission. This proves that such networks can be integrated on FPGA structures. However, this implementation requires a lot of engineering time. In fact, Mission Control even developed their own deployment toolkit for this purpose.

Very recently, Mission Control has started developing MoonNet, a deep-learning-based algorithm to segment rocks, craters, and slopes [3]. Although not much information is shared on the actual characteristics of the network, the training method is elaborated upon. First of all, Mission Control's own Moonyard was used to replicate the lunar environment (see Figure 2.9 [3]), after which the network was finetuned with images from the Chang'E missions [63]. It should be noted that the labeling of this data is a very time-expensive endeavor; in this case, a total of 1800 images were labeled by Mission Control AI-experts. The network was designed for operation on Ispace' HAKUTO-R lander [64] and Rashid rover [65], which crash-landed into the Moon in 2023. The software on board the mission was developed such that it could be updated after retraining on real data acquired during the mission [3], this included data being downlinked to Earth, labeling by experts, and retraining on Earth. The network was deployed onto a Xiphos Q7s, and ran entirely on the CPU. No information on inference time was made public.

In summary, hazard detection methods for Martian and Lunar applications have developed to high accuracy, especially since the emergence of CNNs. Several promising CNN architectures have been described, that could be trained for hazard detection purposes on the Moon. The MultiResUnet architecture emerged as a promising option due to its compact network structure, high accuracy levels, and ability to analyze features at multiple scales. This capability makes it suitable not only for hazard detection but also for monocular depth estimation, as discussed in subsection 2.2.2. Finally, some actual implementations of hazard detection algorithms on the Moon were discussed. Although training meth-

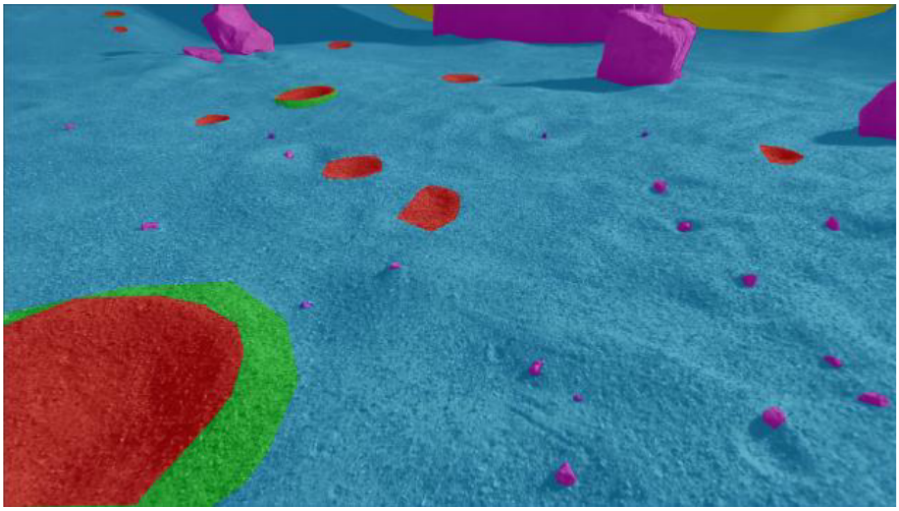


Figure 2.9: An example of a labelled image for training of MoonNet, Mission Control's CNN for segmentation of lunar terrain into different hazard types [3]

ods and some implementation details were shared, the exact network architectures or performance details of the network are not made public.

2.2.2. Monocular Depth Estimation

Monocular depth estimation is the process of determining the distance of objects from a single camera image. It uses computational methods, often leveraging machine learning, to infer the depth information that is typically obtained from stereo vision or multiple cameras. Recent advancements [66] have shown that monocular depth estimation methods are an actual possibility for currently designed missions. Monocular depth estimation methods use architectures like CNNs or Encoder-Decoders [66]. On a low-swap rover like Lunar Zebro, it can be interesting to use semantic information that is already extracted by the segmentation CNN. Very recent work of Wang and Piao [5] represents a notable breakthrough in this area. Their approach, as detailed in their 2023 paper, integrates semantic segmentation with depth estimation in a single CNN. The key addition to the CNN in their method is the multi-scale feature fusion mechanism, which enhances both local and global feature extraction, crucial for accurate depth perception.

Wang and Piao's method [5], by only needing a single image, simplifies the hardware requirements and reduces computational complexity. However, the trade-off is in the accuracy of absolute distance measurements, which are more challenging to obtain from a single image.

To achieve absolute distance determination on lunar surfaces using Wang and Piao's method, training on 'moon yard'-images would be essential, as the model would need to learn from images with actual distances indicated. Until then, the model would only be able to estimate relative distances. Constructing such an environment would be a significant investment for organizations like Lunar Zebro. On the other hand, training a rover only with relative distances can lead to problems like collision risk and inefficient path planning. Wang and Piao indicate that without a large quantity of training data, high-precision depth estimation is not possible [5].

The model developed in Wang and Piao's work also benefits from a multi-path architecture [5], similar to MultiResUnet [4]. Importantly, their model employs a shared encoder for both semantic segmentation and depth estimation [5]. This means that certain layers and parameters in the CNN are optimized to perform both tasks, increasing efficiency and reducing computational load.

In summary, while monocular depth estimation methods, particularly those developed by Wang and Piao employing a multi-path network architecture like MultiResUNet, show considerable potential, their implementation for Lunar Zebro may be prohibitively expensive. This is due to the necessity for a

comprehensive dataset encompassing distances to various lunar features such as rocks and craters. Consequently, both designs incorporating and excluding the monocular depth estimation method will be evaluated in the analysis of the OBCA.

2.3. The Lunar Environment

The Moon harbors many challenges for a small rover like Lunar Zebro. The lunar radiation environment requires special attention for COTS electronic parts. Understanding of the terrain and dust are of importance for the design of a suitable hazard detection algorithm. Finally, the potential application of the swarm on the Moon is important to understand the potential requirements this has on the swarm design and therefore OBCA architecture.

2.3.1. Radiation

In the (cis-)lunar environment one has to deal with radiation from different sources: Galactic Cosmic Rays (GCRs), solar wind & Solar Particle Events (SPEs). The Van Allen belts [67] are prevalent in near-Earth space. This is different from the radiation conditions in LEO, the type of orbit Ubotica Technologies usually works with. This close to Earth, the Earth's magnetic field works as a radiation shield, especially against low-energy particles. Earth's magnetic field filters out most of the radiation due to GCRs and SPEs [68].

GCRs originate from outside the Solar System, likely from explosive events such as supernovae [69]. Due to the high-speed travels through the galaxy, these nuclei lost their surrounding electrons. GCRs path are affected by magnetic fields. Since the Moon lacks a magnetic field, the Moon depends on solar winds to "carry magnetic fields" along with them. The Sun's solar wind cycle therefore determines the intensity of GCR radiation [70]. The annual exposure on the surface of the Moon can go up to 380 mSv during solar minimum [71].

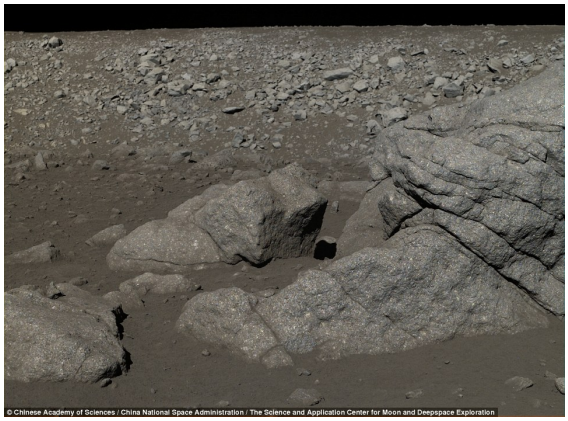
The solar wind is a flow of charged particles which is relatively constant. The solar wind consists of primarily protons and electrons which have a velocity between 200-800 [km/sec]. It has been demonstrated that the solar wind can be readily shielded [67]. However, for SPEs this is less easy. SPEs are eruptions of charged particles (Hydrogen and Helium) due to either Coronal Mass Ejections (CMEs) or solar flares, with energies 3-4 orders of magnitude higher than for solar winds [67]. Doses of more than 1 Sv can occur during cruise to the Moon [71]. SPEs can be predicted, however, they can hurtful to equipment due to the high amount of particles (4-8 orders of magnitude higher per event than GCRs on an annual basis). However, shielding possibilities exist [67].

The Van Allen belts are two layers of charged particles around the Earth held in position by Earth's magnetic field [72]. They extend from around 1000 to 60000 [km] above the Earth's surface. The inner belt consists of protons and electrons, while the outer belt is formed by energetic electrons [72]. The rover will only spend a very short period of time in the belts on its' way to the Moon, which leads to radiation doses around 0.16 Sv [73].

2.3.2. Terrain and Dust

Small and large rocks cover the surface of the Moon as can be seen in Figure 2.10 [74], the rest of the Moon is covered in regolith. This is a layer of powdery soil. Where the lunar mares are covered in about 4-5 meters of regolith, the highland regions are covered in about 10-15 meters of regolith [75]. The dusty underground brings with it serious system safety hazards, such as reduced traction and entrapment of the rover's legs in the dust. However, the locomotion system of Lunar Zebro is developed in such a way, that it does not sink into the regolith layer of the Moon [21].

(a) Images by Chang'e-3 where small rocks litter the lunar surface



(b) The Moon also harbors much larger rock formations

Figure 2.10: Different images of the Lunar surface [74]

Besides rocks, craters form a potential hazard for Lunar Zebro. With a ground clearance of about $40\,\mathrm{mm}$, any slope or rock larger than $30\,\mathrm{mm}$ can be seen as a potential threat. Using rock and crater abundance models [76], as well as rock densities of earlier lunar and martian missions, Manteaux [39] estimated that potential landing regions (which are not set for the mission yet) for the first-generation Lunar Zebro would harbour up to 35 obstacles per 5x5 m^2. His estimations assumed Lunar Zebro would operate in a non-dense rock region (rock abundance less than $5\,\%$), as it is not of interest for the first generation to operate in a rocky region. The rock abundance is measured as the area covered in rocks as a percentage of the total area. Although Lunar Zebro will still not be able transverse landscapes with high rock and crater density, the second generation of Lunar Zebro might be more invested in studying rock compositions as will be discussed in the following subsection, which might be a reason to operate in regions with slightly more rocks.

A study by Bandfield et al. [77], utilizing Lunar Reconnaissance Orbiter Diviner Radiometer data, reveals a predominantly rock-free lunar surface, with the global average rock concentration within the ±60° latitude range being a mere 0.004. Even in regions with higher rock presence, such as maria and highlands, the majority of surfaces exhibit less than 10% rock coverage. Notably, 99.995% of the lunar surface shows rock concentrations below 0.20, and no areas exceed 0.70. This underscores the overwhelmingly low rock abundance across the Moon's surface, providing critical insights for understanding lunar geology and aiding future exploration planning. A maximum rock abundance of $10\,\%$ shall be assumed for the next generation of Lunar Zebro, thus allowing for exploration in almost every region on the Moon.

2.3.3. Potential Resources

The Moon presents a trove of valuable resources, including rare-earth minerals, platinum-group minerals, Helium-3, titanium, volatiles like water, and metals such as aluminum, iron, and platinum [78]. These materials, distributed across the lunar surface, hold immense significance for potential exploitation due to their utility in various industrial and scientific applications. For instance, Helium-3, though not as abundant as once thought, is sought after for nuclear fusion due to its high energy yield and minimal radioactive waste [78]. Water, crucial for sustaining life beyond Earth and serving as a source of oxygen and rocket propellant, has been detected in regions like the Shackleton Crater near the South Pole [78]. However, while we possess approximations of these material concentrations, precise locations remain elusive. This gap underscores the potential role of small rovers, akin to the Lunar Zebro, in exploring and pinpointing exact high-concentration zones.

2.3.4. Lunar Zebro Next-generation Mission Objective

The next generation of Lunar Zebro rovers will primarily be intended for exploration, focusing on investigating unexplored and uncertain lunar regions, rather than mining. These rovers, designed for scouting and analyzing geological features, will concentrate on areas like high latitudes that might pre-

serve volatiles. A significant objective is to search for water and other resources, and measure radiation levels to aid future manned missions. However, they are not equipped to endure extreme conditions found in areas like Lunar Cold Traps (LCTs) [79], limiting their operational scope to less hostile lunar environments. This mission objective aims to enhance our understanding of the Moon's geological composition and identify resource-rich zones.

2.4. (Space) Embedded Systems

The design of an or a given mission depends on the radiation environment, the mission purpose and operational requirements. There are many options to handle the radiation environment, and many types of hardware to choose from when designing such an OBCA.

2.4.1. Radiation effects

Total Ionizing Dose

The Total Ionizing Dose (TID) is the collective name for the accumulation of ionizing radiation in electronics. How much ionizing radiation is absorbed by a component in a satellite or lunar rover depends on different factors such as the duration of the mission, the location in space and the chosen orbit, the location of the part within the satellite, and the relevant shielding around it [80]. The accumulation of ionizing radiation on electronic components can result in TID, which causes degradation of the transistors. TID can significantly alter important characteristics of the transistors, including the threshold voltage shift [80].

Single-event Effects

Single Event Effectss (SEEs) are caused by the interaction of high-energy particles with the sensitive volumes of electronic devices. SEEs can be classified into two types: hard errors and soft errors [80].

Hard errors are characterized by irreversible damage to the device. One common type of hard error is Single-event Latch-up (SEL), which occurs when a parasitic thyristor in a Complementary Metal-Oxide-Semiconductor (CMOS) structure is turned on, leading to a significant increase in current consumption. In some instances, functionality might be only partially lost, but most often the damage is permanent and requires a power cycle to reset the device if recovery is possible at all. [81, 80].
Soft errors are characterized by transient effects that are reversible without causing permanent damage. SEU occurs when a memory cell changes values due to radiation, and the error persists until the memory cell is reprogrammed. Single-event Transient (SET) leads to incorrect values in a user flip-flop, which can be difficult to detect as it may be interpreted as a measurement error. Single-event Functional Interrupt (SEFI) occurs as a result of SEUs or SETs, causing the component to malfunction until it is reset [80].

2.4.2. Fault-tolerant Design Techniques

Redundancy measures can be categorized into four types: hardware, information, time, and software.

Hardware redundancy involves duplicating components to create fail-safe systems. Different forms of hardware redundancy are used in OBCs, such as centralized, cold standby, warm standby, and N-modular redundant OBCs. Each form offers varying levels of internal redundancy and operational readiness, with trade-offs in terms of power consumption and response time to failures. In centralized OBCs, a single internally redundant unit is used, while cold standby involves a backup OBC that is activated upon the main OBC's failure. Warm standby systems keep both main and backup OBCs active but in different operational states. N-modular redundancy utilizes multiple active OBCs running in parallel, with a majority voting system to determine fault handling [82].

Besides hardware redundancy, there are also other options to create redundancy onboard the rover. *Information redundancy* relies on FDIR-methods to ensure that data is transferred accurately across platforms and continuously backed up to prevent data loss during a system breakdown. *Time redundancy* is achieved through the repeated execution of a process on hardware, which aids in the identification of errors and enhances reliability over time. *Software redundancy*, on the other hand, involves running a backup software that performs the same task as the primary program, ensuring that in the event of a fault with the primary software, the redundant software can take over quickly [83].

2.4.3. Hardware

In this section, different architectures will be outlined that could be part of the computing architecture of Lunar Zebro. A focus is on devices that are specialized in processing visual information since the pro-

cessing of the SHRIMP images is the most demanding task for Lunar Zebro's computing architecture. First more traditional architectures like CPUs and Graphics Processing Units (GPUs) will be discussed, after which more specialized devices like Vision Processing Units (VPUs) and Tensor Processing Units (TPUs) will be dealt with. A taxonomy based on the work of Cob-Parro et al. and Ali et al. is presented [7, 8].

Central Processing Unit (CPU)

CPUs are the central part of every computing architecture. They are responsible for simple mathematical operations, searching for instructions, and housing different types of memory, such as RAM, Read-Only Memory (ROM), and cache [7]. While CPUs are very suitable for general-purpose operations but do not have many parallel cores, meaning that parallelization to a high degree is not possible [7]. This makes CPUs less suited for efficient execution of machine learning algorithms. When integrated on a larger system, the CPU will always be integrated with components like memory, peripherals, and interfaces to form a Single-Board Computer (SBC).

Graphical Processing Unit (GPU)

GPUs are devices specially made for operations that require a high level of parallelization [7]. As outlined in Figure 2.15 [84], GPUs features a multitude of cores optimized for parallel processing, allowing simultaneous execution of tasks [7]. GPUs do have a relatively high power consumption [8], possibly hindering their use on a low-SWaP spacecraft.

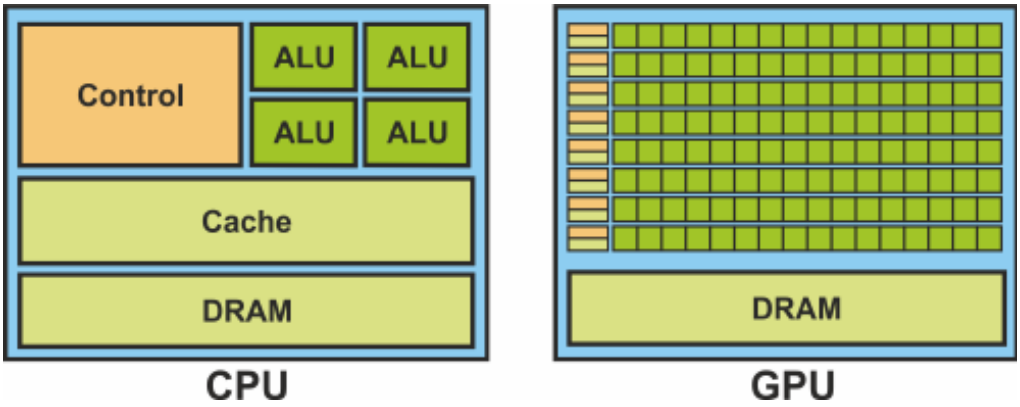


Figure 2.11: Comparison of CPU versus GPU architecture [84]

FPGAs

FPGAs are reconfigurable and general-purpose [85]. The programmable logic blocks can be reprogrammed to efficiently perform the tasks of the user. However, reprogramming FPGAs can be complex and thus a costly process.

A range of studies have explored the implementation of CNNs on FPGA. Bahl et al. developed and implemented CNNs for semantic segmentation on a basic FPGA [86]. Voroshazi [87] and Perko [88] both focused on the design and implementation of CNN units on FPGA, with Voroshazi extending the architecture with a soft processor core [87] and Perko achieving high performance with a low-cost simulator [88]. Wei introduced a systolic array architecture for high throughput CNN inference [89], while Phu et al. designed a configurable CNN core generator for FPGA, achieving high accuracy and low error rates [90]. These studies collectively demonstrate the potential of FPGA-based CNN implementations for high performance, low power usage, and reconfigurability.

FPGAs are available in two architectures: SoC FPGAs and standard FPGAs [91]. SoC FPGAs combine both FPGA and CPU on a single chip, often incorporating ARM processors in edge scenarios, as depicted in Figure 2.12 [91]. This integrated design offers higher internal bandwidth between the

CPU and FPGA while decreasing the size of the chip and the power demand [91]. In contrast, standard FPGAs have separate FPGA and CPU components, while decreasing the size of the chip and the power demand [91]. While both types work with their own external memory and can access each other's memory through internal connections, the SoC FPGA's integrated design typically results in higher performance efficiency due to reduced latency and better communication between the CPU and FPGA components, while fitting on a compacter chip [91].

The Xilinx Xiphos Q7S [30], the computer used for the first Lunar Zebro design is indeed a SoC FPGA.

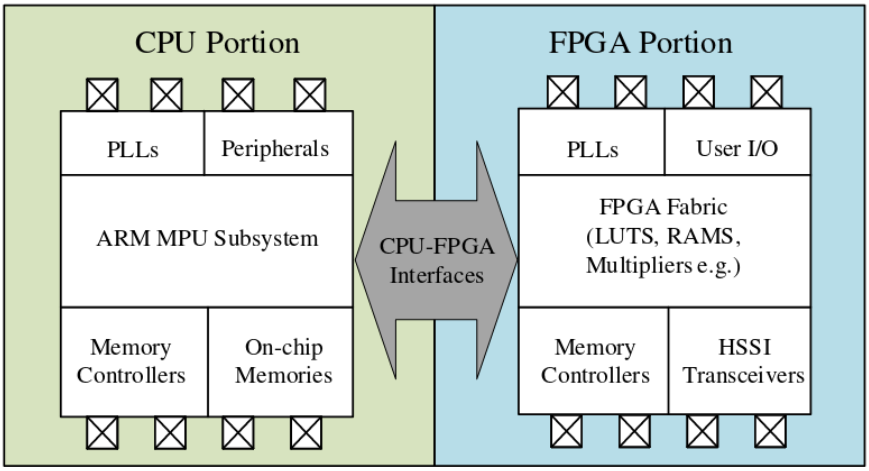


Figure 2.12: A SoC FPGA [91]

Application Specific Integrated Circuits (ASICs) & TPUs

ASICs are pre-designed, application-specific devices, that are optimised for one very specific task [8]. Apart from more mass-produced ASICs like the Google TPUs they are often expensive alternatives, as they are produced in low volume.

TPUs are ASICs, designed specifically for machine learning (TensorFlow operations), excel in tasks involving extensive matrix multiplication, outperforming general-purpose processors in applications like image and speech recognition, natural language processing, and recommendation systems. However, their use is limited to specific types of neural networks [8]. The TPU architecture, as depicted in Figure 2.13 [92], is similar to the simple CPU architecture, but a matrix multiplication system is added [7]. To serve this matrix multiplication unit, high-speed data pathways are put into place. The systolic data array is a grid of arithmetic units fit for parallel operations [92]. Google's TPUs have been introduced to space applications for the first time through the SpaceCube Edge TPU SmallSat Card [6]. However, Goodwill et al. [93] state that "this card is designed to be monitored by a complementary high-performance processor which is responsible for powering on/off the individual Edge TPU modules".

VPUs

VPUs specialize in processing visual data with the help of CNNs [94]. These devices are specialized in the matrix convolutions that stand at the base of every deep-learning algorithm [7]. The VPU architecture is depicted in Figure 2.14 [95]. The architecture does not include the same attributes as the CPU, showing that the VPU is intended to be added to a general computing device, taking convolutional computations away from it [7]. The architecture in Figure 2.14 [95] shows the parallel Streaming Hybrid Architecture Vector Engine cores (SHAVEs) of the Myriad X, allowing for parallelization of image channels or even applications. The SHAVEs are interfaced with hardware accelerators designed to execute machine vision algorithms both efficiently and rapidly [7].
Ubotica Technologies [10], intent on enabling complex visual operations directly on satellites, has opted for Intel's Myriad X VPU [12] to power these advancements. In pursuit of this goal, they have devel-

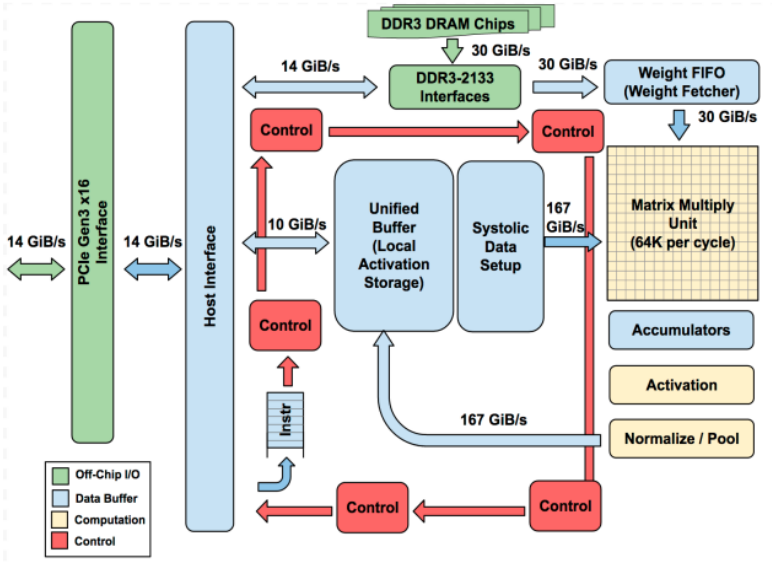


Figure 2.13: General schematic of a TPU architecture [92]

oped the CogniSat XE2 board [11], a specialized platform that effectively leverages the VPU's capabilities, but is thus far only proven in LEO environments. The XE2 board is depicted in Figure 2.15 [11]. The CogniSat XE2's compact form factor, with dimensions aligning with the PC/104 standard [96] (99x98x18.5mm) and a mass of just 80 grams, ensures that it meets the stringent payload requirements of space missions.

AI-accelerating methods

In the field of AI acceleration, there are many methods to increase the efficiency or time consumption of machine learning algorithms. It is essential to comprehend the following key terms and concepts:

- *Quantization* is a method that reduces the precision of numerical data used in computations [97]. It involves transforming floating-point representations into lower-bandwidth formats, such as 16-bit or 8-bit integers. This process is crucial in deep learning models with millions of parameters, as it substantially decreases the computational demands and memory usage. The primary advantage of quantization is its ability to make models more lightweight and faster, facilitating their deployment on devices with limited processing capabilities. However, it requires careful implementation to ensure that the reduction in data precision does not significantly impact the model's accuracy or performance.
- *Parallelization* takes advantage of the architecture of modern processors which are capable of performing many calculations simultaneously [97]. This method is effective for tasks that can be divided into smaller, independent units of work. In AI, parallelization is extensively applied in the training and inference stages of neural network models. By distributing computations across multiple cores or processors, parallelization greatly accelerates processing times, making it an essential technique for handling large-scale neural networks and complex AI algorithms. This approach not only speeds up computations but also allows for more efficient utilization of available hardware resources [97].

Other methods are the reuse of data and pruning of neural networks [91].

2.4.4. Hardware in Deep-Space & COTS Solutions

The NewSpace industry [18] that is arising introduces the use of COTS hardware in space systems. NASA defines COTS as: "An assembly or part designed for commercial applications for which the item manufacturer or vendor solely establishes and controls the specifications for performance, configuration, and reliability (including design, materials, processes, and testing) without additional requirements

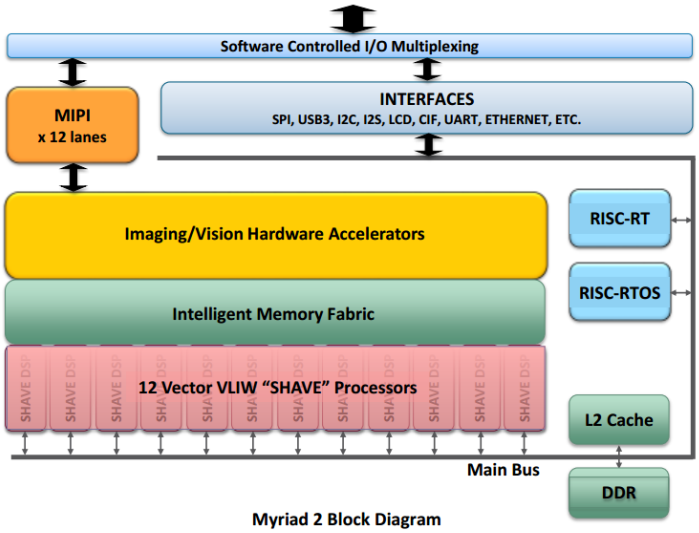


Figure 2.14: General schematic of a VPU architecture [95]

imposed by users and external organizations" [98].

Despite their low cost, COTS hardware is infrequently used in space exploration missions like Lunar Zebro. This section delves into the application of COTS technology in missions beyond Earth's orbit, with a particular emphasis on low-SWaP systems and rovers. Various missions are examined to understand the SOTA for OBCAs in deep space missions. Especially where and how COTS solutions have been implemented, aiming to provide a clear overview of their usage in the context of deep-space exploration.

For missions in deep space, one can refer to Lunar and Martian rovers. Although the rovers of the Chang'e 3 and 4 (Yutu 1 and Yutu 2) missions are very interesting for comparison, no details are shared about the design of their respective OBCs by Chinese National Space Administration (CNSA). The same is true for the Chandrayaan-2 rover, Pragyan. The Luna 28 mission is the only Luna mission with a rover. However, information about this mission is not yet disclosed [99]. Similarly, this applies to NASA's CADRE mission, a lunar swarming mission with low-SWaP rovers [100] that shows great resemblance with the next generation of Lunar Zebro.

NASA's Martian rovers, despite their larger size compared to the Lunar Zebro, utilize onboard computers with limited capabilities due to Mars' harsh radiation environment and the high cost of rad-hardened devices. The Spirit and Opportunity rovers both used an OBC with similar specifications: a 20-MHz RAD6000 CPU [101] with 128 Mbytes of RAM, 256 Mbytes of flash memory, and 0.25 Mbytes of EEPROM [102]. The runtime for Rockster, the hazard detection algorithm on board was 600-900 seconds, on a 1Kx1K-pixel image [53]. This RAD6000 and its accompanying board are rated somewhere between 200,000 and 300,000 dollars [103], which is out of budget for a mission like Lunar Zebro.
Curiosity and Perseverance have the same board specifications: a 200 MHz RAD750 core, 256 MB RAM, 2048 MB Flash, and 0.25 MB EEPROM [102]. In comparison, the computing power of the Curiosity and Perseverance rovers is significantly lower than that of the original Apple Watch. The Apple Watch boasts 512MB of RAM, 8GB of storage, and a 520MHz processor, which far surpasses the capabilities of the rovers' computer systems [104].
This is in stark contrast with COTS devices like Lunar Zebro's Q7s OBC with dual-core processor up to 766 MHz, 768MB RAM [30].

Ingenuity is a helicopter, specially designed to fly in Martian conditions (1% of Earth's atmosphere and 38% of Earth's gravity) that accompanied the Mars 2020 Perseverance Rover as a low-SWaP terrain scout [105].
The helicopter successfully performed 52 flights so far, where it had only 5 flights planned [106]. Unique to Mars missions, NASAs Jet Propulsion Laboratory (JPL) chose to adopt a COTS SoC, namely Qual-

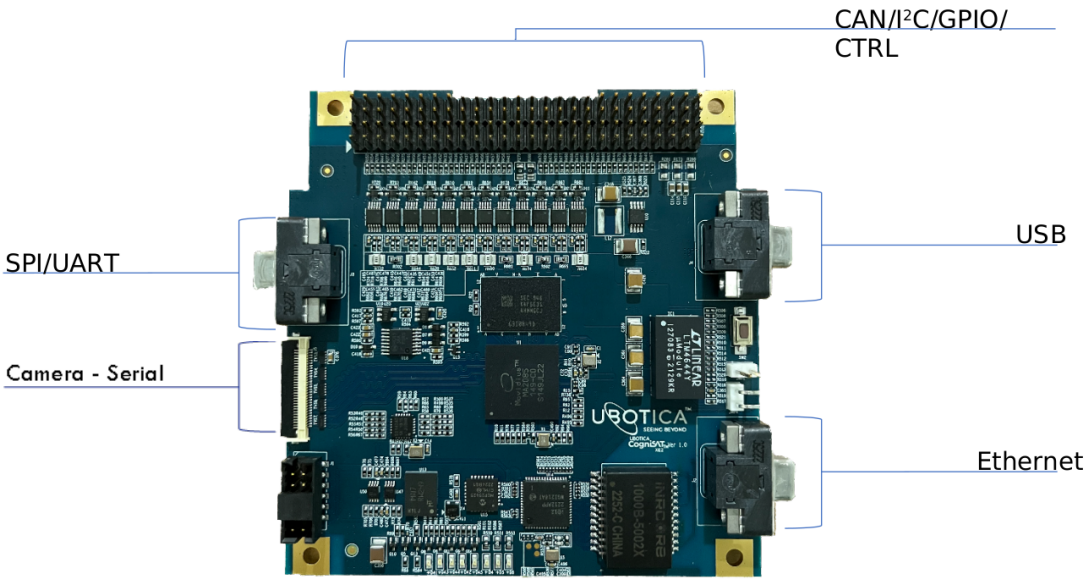


Figure 2.15: The CogniSat XE2 board with its most important features highlighted [11]

comm's Robotics Flight 801 platform [107, 108]. This chip is an adaptation of Qualcomm's original Snapdragon to allow for functionalities fit for flight. NASA chose this board specifically for its relatively low power consumption while allowing for the exact capabilities necessary for a successful flight on Mars. Moreover, this type of board are in a price range more realistic to the Lunar Zebro mission, staying within a thousand dollars [109]. Interestingly, due to the required real-time action of the drone, the CPU on board needs to be more powerful (2.26 GHz) than that on board the Perseverance rover. Lunar Zebro will need to make 'real-time' decisions but will have more time than the drone, mostly because it moves much slower.
Moreover, although the inference times and energy consumption of the Snapdragon GPU, Neural Processing Unit (NPU) and DSP are comparable or even faster to the Myriad X for a deep learning-based classifier [110], the OBC of the Ingenuity helicopter was probably not as constrained for energy, as flights were only $90\,\mathrm{s}$ [106]. Snapdragon SoC devices like the 801 flown on Ingenuity [106] are optimised for graphic performance and inference speed [111]. Compared to the CogniSat XE2 with the Myriad X VPU, the Snapdragon with its CPU, GPU and DSP, is more general-purpose, which would normally go at the cost of energy-efficiency of tasks that the Myriad X is specialized for. For example, the Adreno GPU on the Snapdragon SoC that was flown on Ingenuity was designed for 'premium graphics and console-quality gaming' [111].

Cross et al. demonstrated that MoonNet (discussed in section 2.2) could run entirely on the PS of the Q7S, although the limited computational power and memory were a challenge [3]. Also, engineers from the same company, Mission Control, ran an earlier version of MoonNet, the Machine Learning-based Autonomous Soil Assessment System (ASAS) was implemented on the Q7S and Q8S [112]. In these demonstrations, the FPGA devices would not function as the main computing architecture, but rather as a payload [3].

From the survey in this section it has become apparent that extraterrestrial applications of COTS OBCs are rare. Especially for small, extraterrestrial missions like Lunar Zebro, not much research has been done. Especially not when looking into more advanced capabilities like swarming, which will be elaborated upon in the coming chapters. George and Wilson [113] even state that 'the viability of these autonomous applications in terrestrial applications relies on large, distributed systems with GPUs, which is infeasible on a space system without improved next-generation computing devices and optimized platform applications'.

Adding such hardware to an extraterrestrial nano-rover has not been researched before, therefore leaving a research gap.

2.4.5. Computing Architectures

Different computing architectures can be utilized to perform the tasks on board the lunar rover. 'Computing architecture' here refers to the organization of key components of the computer, including memory systems and the paths for data and instructions to and from the processor.

The von Neumann architecture [114] integrates both data and program instructions in the same memory space. This unified approach simplifies programming and system design. Central to this architecture is a sequential processing mechanism, where instructions are executed one after another. However, this sequentiality leads to a limitation known as the 'von Neumann bottleneck.' This bottleneck arises because the processor often has to wait for data and instructions to be delivered from the memory, potentially slowing down the overall processing speed, especially as processor speeds have outpaced memory speeds. The architecture's simplicity has made it a dominant design in computing [114], but this bottleneck highlights the need for innovations to enhance data throughput and processing efficiency for more advanced space missions.

The parallel von Neumann computing architecture, utilizing multiple processors for concurrent processing, is advantageous for complex tasks such as real-time data analysis and autonomous decision-making [115]. Duncan [115] differentiated between synchronous architectures and Multiple Instruction, Multiple Data (MIMD) architectures. Synchronous architectures are systems where all the processors or components work together at the same speed, controlled by a single central clock. This means that every part of the system follows the same timing for processing tasks, ensuring that everything stays in sync. In contrast, MIMD architectures allow each processor to execute different instructions on different data simultaneously, offering more flexibility and scalability for diverse computational tasks.

The Harvard architecture presents an alternative to the von Neumann architecture [116], with its separate paths for data and instructions, facilitating simultaneous access and thus enhancing processing speeds. The separate pathways for data and instructions in the Harvard architecture mean that data can be processed and instructions fetched simultaneously, significantly speeding up computations needed in real-time signal processing applications.

Emerging architectures like neuromorphic computing [117], which mimics the human brain, and quantum computing [118], based on quantum mechanics, represent possible approaches for the unforeseeable future. At the current moment in time, these technologies are too immature.

For the Lunar Zebro's OBC, while different architectures have their merits, the need for parallel processing is clear, as the rover will have to run parallel processes. In this scenario, architectures capable of facilitating parallel processing, including parallel von Neumann, MIMD, and others, can be effective. The key requirement is not the separation of data and instructions in different memories, as seen in the Harvard architecture, but the ability of the architecture to handle multiple processes concurrently. This is reflected in requirement **LZ-OBCA-DES-006**.

2.5. Swarming

For the next generation of Lunar Zebro (see Figure 4.10) [119], incorporating swarming capabilities is an important goal. This approach involves autonomous robots working collaboratively towards common objectives to increase robustness, flexibility and scalability at lower cost [120, 121, 122].

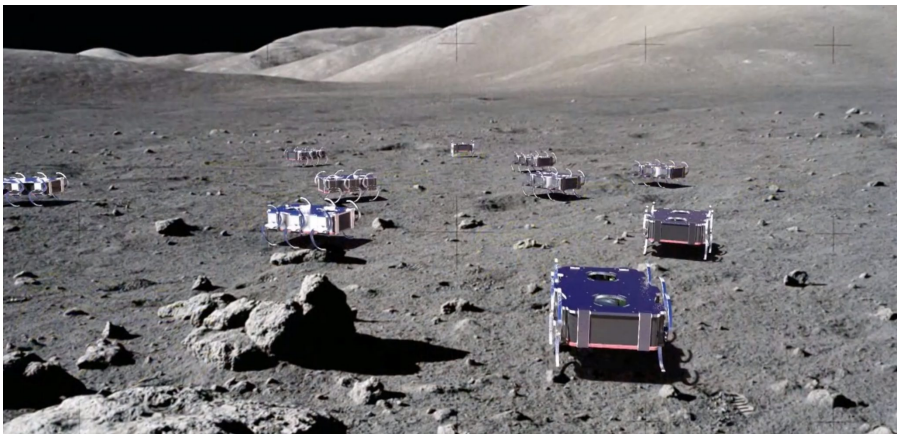


Figure 2.16: A render of a swarm of Lunar Zebros on the Moon [119]

An earlier example of swarm robotics in space includes the Autonomous Nano-Technology Swarm (ANTS) mission, which consisted of a fleet of miniaturized, autonomous spacecraft to explore the asteroid belt [123]. Recently, NASA started developing the Cooperative Autonomous Distributed Robotic Exploration (CADRE) mission [124]. Similar to Lunar Zebro, CADRE involves a network of small rovers to the Moon, designed to operate autonomously and communicate via a mesh network. These rovers are equipped with two stereo cameras, navigation sensors, and a multistatic ground-penetrating radar [124]. Not much information about this mission is made public; the exact swarming methods and computing architecture design cannot be found publically.

For the software design and the requirements this poses on the computing architecture of Lunar Zebro, the required rules and software for swarm robotics should be outlined. Hamann and Wörn describe that individual rovers make decisions based on locally gathered information and simple rules, including moving towards or away from neighbors, aligning with neighbors' orientation, or aggregating around a specific point of interest [125]. Although simple rules steer the individual, complex behavior emerges for the swarm [126]. This decentralized approach ensures adaptability to environmental or swarm changes and makes the swarm scalable. The computational demands for the individual are minimal due to the simple rules. Therefore, the swarming software will not pose heavy requirements on the processing power of the rover's computing architecture. This will be further discussed in subsection 4.3.7.

3

Requirements Engineering & Conceptual Design

This chapter describes the conceptual design of the OBCA architecture for the next generation of Lunar Zebro. A V Model systems engineering approach was used during this research project. The chapter is divided into two sections. Section section 3.1 involves the first system engineering steps, including the identification of stakeholder and system requirements. Moreover, the functional flow of the OBCA tasks is conceived, and technical budgets are determined, to be able to set SMART requirements [127]. The second section outlines the different considered design options, and discusses the final choice of design options considered for further testing.

3.1. On-board Computing Architecture Systems Engineering

For the design of the OBCA a V (”Vee”) model approach was used, as initiated by Forsberg and Mooz [128] and later developed in the International Council on Systems Engineering (INCOSE) Systems Engineering Handbook [129].

For this project, the model needs a slight adaptation. Because of the choice for COTS boards, the subsystem level can be wavered. The system requirements describe the OBCA to a level of sufficient detail. The adapted version of the Vee model is displayed in Figure 3.1. The modified Vee model, as depicted in Figure 3.1, emphasizes an iterative design methodology. This approach facilitates continuous refinement through successive stages of development, testing, and validation. The final validation stages for the OBCA, with all algorithms implemented on the rover, do not fit within the scope of this project. This will be reflected upon in chapter 8.

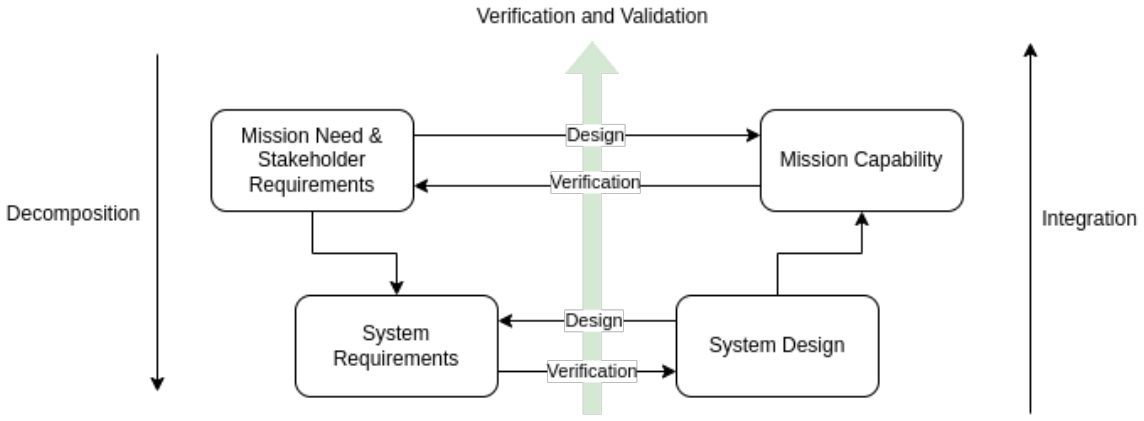


Figure 3.1: The ”Vee” model modified for this project. The subsystem level is left out. Inspired by [129].

3.1.1. Need and Mission Statement

The initial step involves identifying the problem that requires a solution and specifying the need that the system is intended to fulfill, a process facilitated through the creation of a 'Need Statement.' The Need Statement for the OBCA is formulated as follows:

> *There is a need to run DL-based Hazard Detection on-board Lunar Zebro, a SWaP-constrained lunar rover, designed to explore a given area on the Moon in the form of a swarm of likewise individuals.*

The mission statement explains how the 'need' above will be satisfied:

> *Design a cost-effective OBCA for the Lunar Zebro rover to enable exploration of the Moon, supporting DL-based hazard detection, swarming, DL-based FDIR, and autonomous mission planning, by taking advantage of COTS technology.*

3.1.2. Stakeholders

To be able to understand which stakeholders are involved and what the system boundaries of the mission are, the mission context diagram in Figure 3.2. The Lunar Zebro system can be said to consist of one individual rover itself, including its payloads. These payloads interact with the lunar environment, either to recognize where to walk or to leverage the scientific data as part of its mission. The individual rover might be constrained by requirements from the launch provider and might be in contact with a lunar base or lander during the mission, through which eventually the Mission Operations Center shall be able to influence the mission if necessary. Finally, other rovers will also be part of this Lunar Zebro mission, which is what makes this next generation of Lunar Zebro unique. Finally, the TU Delft, the Lunar Zebro team and space agencies will use the data obtained by the Lunar Zebro mission.

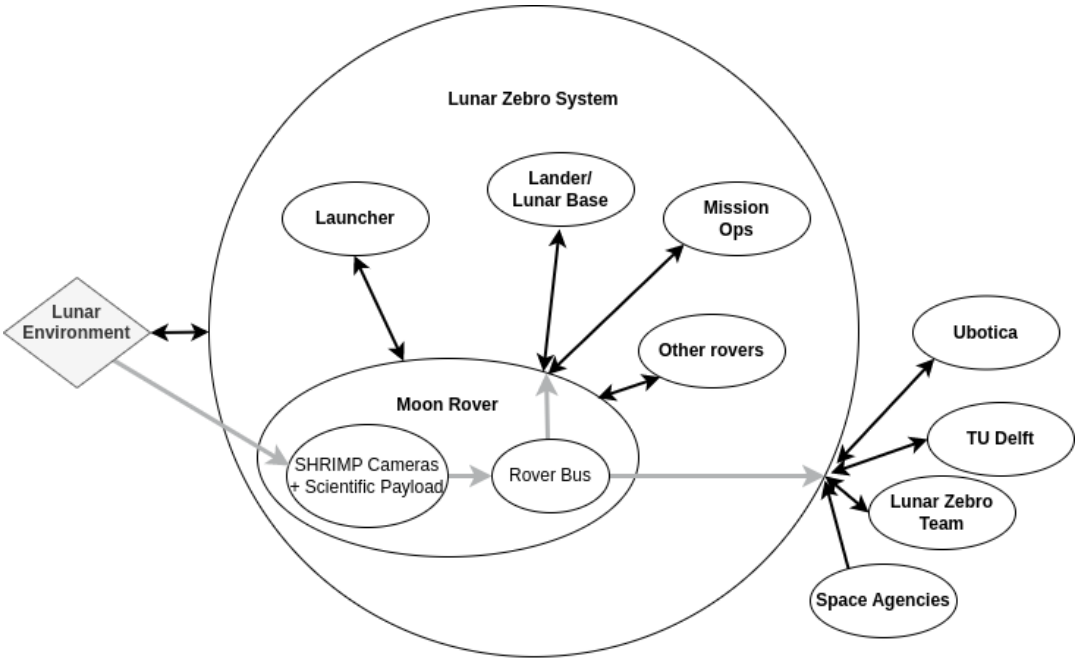


Figure 3.2: Mission context diagram for the next-generation Lunar Zebro mission

The different stakeholders are summarized in Table 3.1.

Table 3.1: Stakeholders of the next-generation Lunar Zebro mission

Stakeholder	**Group** Active/ Passive/ Customer	**Rationale**
Lunar Zebro Team	A/C	As the primary customer, they set specific requirements and functional objectives for the OBCA, guiding its development to align with the overall rover design and mission goals.
Launch Provider	A	The launch provider dictates environmental and mechanical constraints for the OBCA, impacting its design specifications. They also influence key mission aspects such as integration procedures and launch timelines.
Lander/Lunar Base	A	Defines communication protocols. Determines data relay and operational coordination methods.
Mission Control	A	Provides minimal overruling and observation for the largely autonomous rovers, impacting requirements for remote command capabilities and data monitoring.
Ubotica Technologies	A	Influences OBCA design by potentially integrating an AI accelerator, impacting computational capabilities, the bus structure and power management strategies.
TU Delft	A	As the organizing university and sponsor, they guide overall project objectives and focus, particularly emphasizing the scientific outcomes and educational aspects of the OBCA design.
Space Agencies	P	Provide funding and mission objectives, influencing OBCA's design priorities to align with broader scientific and exploratory goals.
Testing Facilities	P	Offer environments for validating OBCA's performance and resilience under simulated space conditions, impacting design validation and verification processes.

Having identified all stakeholders, stakeholder requirements were pinpointed to inform and direct the configuration and structure of the OBC. These stakeholder needs will shape decisions concerning the OBC from its design and testing phases through to operational use. The stakeholder needs lead to stakeholder requirements as stated in Table 3.2. These requirements are labeled following the following convention: **LZ-OBCA-SH-[REQUIREMENT NUMBER]**.

Table 3.2: Lunar Zebro OBCA Stakeholder requirements

Type	**Label**	**Requirement**	**Char/Cap**	**Priority**: Essential/ Conditional/ Optional
Stakeholder	SH-001	The OBCA shall be constructed using COTS components.	Char	E
	SH-002	The OBCA shall be designed for easy manufacturing and implementation.	Char	C

Table 3.2: Lunar Zebro OBCA Stakeholder requirements

Type	Label	Requirement	Char/Cap	**Priority**: Essential/ Conditional/ Optional
	SH-003	The OBCA shall consume a maximum of 10% of the total energy budget per operational cycle.	Char	E
	SH-004	The OBCA shall not require a peak power exceeding the constraints of the EPS.	Char	E
	SH-005	The OBCA shall be able to operate in the lunar (radiation) environment.	Cap	E
	SH-006	The OBCA shall withstand specified launch loads and vibrations.	Char	E
	SH-007	The OBCA shall be designed to integrate with and operate on the next-generation Lunar Zebro nano-rover.	Char	E
	SH-008	The OBCA shall be capable of running Deep Learning-based hazard detection algorithms.	Cap	C
	SH-009	The OBCA shall incorporate error detection and correction mechanisms.	Cap	E
	SH-010	The OBCA shall be capable of running DL-based FDIR on subsystem data.	Cap	C
	SH-011	The OCBA shall be able to recover subsystems when a fault is detected.	Char	E
	SH-012	The different subsystems shall be controlled through modular applications.	Char	C
	SH-013	Lunar Zebro shall be able to perform its tasks without direct contact with Earth or a human operator.	Char	E
	SH-014	The OBCA shall support the swarming capability of the next-generation rover.	Char	E
	SH-015	The OBCA shall ensure the rover to be able to move with an average velocity of [2] cm/s.	Cap	C
	SH-016	The OBCA shall use a RS485 serial communication bus to connect with all subsystems.	Char	E
	SH-017	The OBCA shall support scientific measurements with a spectrometer.	Cap	E
	SH-018	The OBCA shall support a Linux Operating System (OS).	Char	E
	SH-019	The data interface between a main OBC and AI-accelerator shall be through Ethernet.	Char	E

All stakeholder requirements except **LZ-OBCA-SH-018** and **LZ-OBCA-SH-019** follow from the background as discussed in chapter 2. The requirement **LZ-OBCA-SH-018** was established to ensure that the OBCA supports a Linux operating system, facilitating compatibility and seamless integration with Lunar Zebro's Linux-based operational infrastructure.
The requirement **SH-019**, stated by Ubotica Technologies, specifies the use of an Ethernet interface for the data connection between the main OBC and AI-accelerator, aligning with the state-of-the-art standards for AI-accelerator integrations in contemporary on-board computer systems.

3.1.3. System Requirements

From the stakeholder requirements follow the system requirements, which are decisive for the design of the OBC. As a rule of thumb, all system requirements shall have a parent stakeholder requirement, for traceability purposes. The requirements are set up following the Specific, Measurable, Attainable, Re-

alistic and Traceable (SMART) criteria for requirements. The requirements are labeled in accordance with the following convention: **LZ-OBCA-[REQUIREMENT CATEGORY]-[REQUIREMENT NUMBER]**. The requirements are split into six categories, namely:

1. **Functional requirements** describe what the system should do, outlining its specific functionalities or operations.
2. **Design requirements** specify how the system will be structured or designed.
3. **Performance requirements** specify how well the system should perform (i.e. processing speed).
4. **Interface requirements** define interactions between the system and other subsystems or components.
5. **Reliability, Availability, Maintainability, and Safety (RAMS) requirements** focus on ensuring the system's reliability, availability, ease of maintenance, and safety in the Moon's environment.
6. **Operational requirements** have to do with the operational cycles of the rover and the OBCA.

The criticality of every requirement is also graded with high, medium or low. This allows for prioritization when required. Moreover, the verification method for every requirement is given following the guidelines for requirement verification by European Cooperation for Space Standardization (ECSS) [130], which is either review by design, analysis, inspection or test. Finally, a short rationale for the requirement is added.

Table 3.3: Lunar Zebro OBC system requirements

Type	Label	Parent	Requirement	Criticality	**Verification Method** (Analysis/ Inspection/ Test/ Test by experiment)	Rationale
Functional	FUN-001	SH-016	The system shall acquire and store [128] kByte of scientific data at a frequency of [0.0016] Hz.	High	A	Assuming 6 measurements per hour with low-mass spectrometer [35].
	FUN-002	SH-008	The system shall enable deep learning-based rock segmentation of rocks larger than [3] cm.	High	T	The rover can overcome obstacles up to 3 cm in height (see section 2.1). Choice for segmentation substantiated in section 2.2.
	FUN-003	SH-013	The system shall autonomously replan its path every operational cycle of [40] seconds.	High	A	
	FUN-004	SH-010	The system shall detect faults in subsystems using deep learning for at least one variable per subsystem.	High	I,A	To ensure subsystem integrity.
	FUN-005	SH-009	The OBCA shall be capable of powering itself ON and OFF.	Medium	I	For energy management and safety.
	FUN-006	SH-011	The OBCA shall possess the capability to reboot all subsystems independently.	Priority	Category	Rationale.
Design	DES-001	SH-004	The system shall have a peak power consumption not exceeding [10] W.	High	I	Determined in Table 3.1.7.
	DES-002	SH-003	The system shall have an average power consumption not exceeding [6.9] watts <TBC>.	High	I,T	Determined in subsection 3.1.7.
	DES-003	SH-006	The system shall be able to withstand loads of [20] N.	High	I	For structural integrity under launch load [131].
	DES-004	SH-006	The system shall withstand vibrations in the frequency range of 20 Hz to 10,000 Hz.	High	I	For durability against vibrations [131].
	DES-005	SH-001	The system shall consist of COTS parts.	High	I	Defined in subsection 2.4.4.
	DES-006	SH-015	The system shall support parallel processing capabilities to handle multiple operational tasks concurrently.	High	I	
Performance	PERF-001	SH-007, SH-008, SH-009, SH-010, SH-014	The system shall have a minimum RAM capacity of [TBD] Bytes, including a 25% buffer.	High	A	Will be determined in chapter 6. Depends on the swarm configuration.
	PERF-002	SH-007, SH-008, SH-009, SH-010, SH-014, SH-017	The system shall provide a non-volatile storage capacity of [500] MB.	High	A	Determined in chapter 6.

Table 3.3: Lunar Zebro OBC system requirements

Type	Label	Parent	Requirement	Criticality	**Verification Method** (Analysis/ Inspection/ Test/ Test by experiment)	Rationale
Interface	INT-001	SYS-014	The system shall provide an RS485 data interface with the SHRIMP camera(s).	High	I	For camera control and data handling.
	INT-002	SH-016	The system shall provide a RS485 serial communication interface with leg module motor drivers.	High	I	Controlling the drivers.
	INT-003	SH-016	The system shall provide a RS485 serial communication interface with communications hardware.	High	I	For communication management and data transmission.
	INT-004	SH-016	The system shall provide a RS485 serial communication interface with the PPU.	High	I	For power distribution and management, also power cycling of subsystems.
	INT-005	SH-007	The system shall be able to operate on a supply voltage of 3.3, 5 or 12 V.	High	I	For compatibility with the specified power supply and power converters.
	INT-006	SH-019	In the case that the XE2 is connected to a central OBC, data shall be transmitted through Ethernet.	Medium	Inspection	Ubotica demand. More space-proven than USB.
RAMS	RAMS-001	SH-005	The system shall be able to withstand TID up to [341] rad in operation.	High		For radiation tolerance in lunar environment.
	RAMS-002	SH-005	The system shall be able to operate in a vacuum.	High	I	
	RAMS-003	SH-005	The system shall have an operational temperature range of [-40] to [80] degrees Celsius.	High	I	
	RAMS-004	SYS-009	The parts of the system shall be tested for latch-ups due to SEEs at [105] MeV of ionizing radiation.	High	I,A	Recommended flux in [112].
	RAMS-005	SH-005	The system shall be able to store three redundant copies of firmware and applications.	High	I,A	Enables implementation of fault-tolerant verification methods.
	RAMS-006	SH-005	The system shall be protected against direct exposure from lunar dust.	High	I	To prevent damage from abrasive lunar dust.
	RAMS-007	SH-007	The system shall function for at least 14 Earth days.	High	I,A	For mission duration alignment with lunar day cycle.
Operational	OPER-001	SH-007	The system shall be in solar-harvesting mode during [4] hours and active for [1.25] hour alternately.	Medium	A	To optimize power usage and system longevity.
	OPER-002	SH-015	The system shall ensure the rover to be able to move with an average speed of [2] cm/s.	Medium	A	To achieve effective ground coverage within operational constraints.

Table 3.3: Lunar Zebro OBC system requirements

Type	Label	Parent	Requirement	Criticality	**Verification Method** (Analysis/ Inspection/ Test/ Test by experiment)	Rationale
	OPER-003	SH-007, SH-009	The system shall leave [1038] DMIPS available for control of the LMS, the Master Control Program, EDAC and PPU control.	Medium	A	To ensure resource allocation for mobility, EDAC, MCP, and PPU, see section 2.1.
	OPER-004	SH-010, SH-013, SH-015	The system shall process new images every [0.8] meters <TBC> of walking distance.	High	T	Ensure operation without collisions.

3.1.4. Operational Details and Assumptions

The current Lunar Zebro design and contemporary rover component technologies form the basis of several assumptions. The average walking speed of the rover is assumed to be 2 cm/s, where the maximum speed of the rover is assumed to be 5 cm/s. Manteaux [39] found that with a simple You Only Look Once (YOLO) algorithm and downsized images (160x160 pixels), all obstacles were detected up to a distance of 0.8 meters. Moreover, Manteaux [39] notes that when the covered distance or cycle time is increased, the planned path becomes less efficient. Especially for a small rover like Lunar Zebro, which is so close to the ground that it cannot oversee large distances. Therefore, 0.8 meters is taken as the distance after which new images need to be analysed. This is a conservative estimate, as improved hazard detection algorithms and the possibility to analyze higher-resolution images will increase the precision. A sensitivity analysis will be performed on this distance in subsection 6.2.2.

Moreover, it is assumed that the rover works in duty cycles of 75-minutes, after which 4 hours are required to charge the batteries. It is assumed housekeeping of each subsystem is shared once per 5 minutes and that two forms of data are available, i.e. Voltage and temperature.

The Master Control Program, the EDAC for the on-board computing architecture and the PPU application run simultaneously with the hazard detection cycle. Knowing that the locomotion app requires 5%, it is assumed that these apps require 10%, 5% and 5% of the computing power of the ARM Cortex A9 dualcore, continuously.

Finally, the rovers are assumed to be able to determine their own positions, relative to the lunar lander or base.

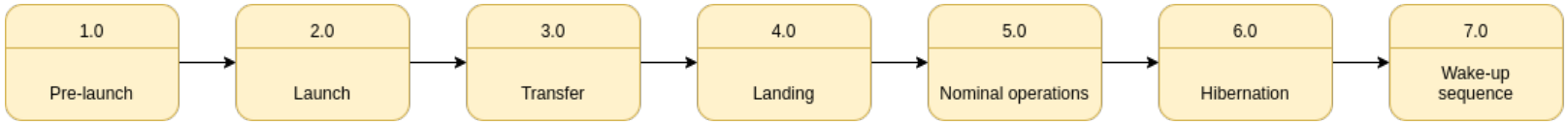
3.1.5. Functional Flow

After having broken down the functions of the OBCA, the order of processes of the rover and its OBCA needs to be outlined, as the on-board process pipeline plays an important role in the design decisions for the OBCA and the algorithms it runs. To start with, in subsection 2.1.3 the different mission phases were already discussed (the pre-launch, launch, Earth orbit, cis-lunar transit, lunar orbit and lunar surface phases, see Figure 3.3a). This lunar surface phase can be further broken up as in Figure 3.3b.

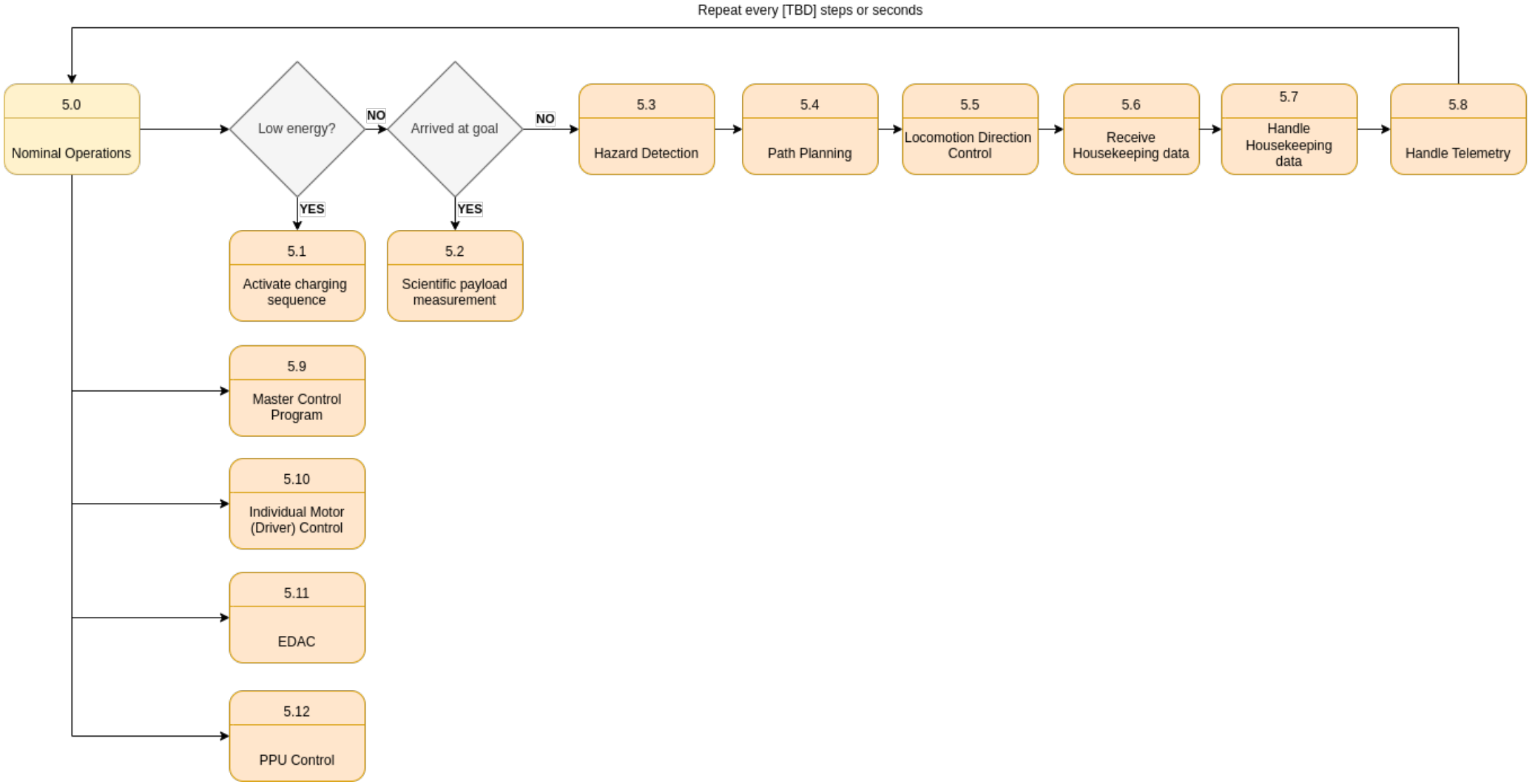
Lunar Zebro's stay on the Moon will be broken up into charging periods and active periods. The charging period and sequence is discussed in earlier work done by the Lunar Zebro team [26]. The active periods will consist of static 'operational cycles'. The length of this operational cycle depends on several factors, such as the average velocity of the rover and the distance up to which accurate hazard detection can be guaranteed. Only when defective behavior of subsystems is detected, will the static cycle be overruled by fault isolation and recovery measures.

During one operational cycle, Lunar Zebro and its OBCA specifically, will need to perform a series of tasks. These tasks are depicted in Figure 3.3b. The operational cycle always starts by determining if there is enough energy for another cycle. If not, the charging sequence is activated. If the rover has enough energy to continue operations for one more cycle, the rover will determine if it has arrived at its goal, as this would require a scientific measurement. If not, the hazard detection pipeline is started. The algorithms involved can determine where in the images taken by SHRIMP, hazards are present. Having detected the hazards, the rover can plan the best available path towards the goal and the directions can be calculated for the locomotion system. Then, housekeeping data from different subsystems is requested and handled by the OBCA.

Finally, in the final period of every operational cycle rovers share their locations, image data, and housekeeping data with one another. This is performed in a predefined schedule so that neighboring rovers receive each other's data as efficiently as possible. The communication sequence could also be performed simultaneously with the other processes, but as the rovers only replan their paths once per cycle, continuous sharing of locations is redundant.



(a) Functional flow diagram of the main phases that the rover goes through throughout the mission lifetime.



(b) Functional flow diagram of the operational cycle of the OBCA of an individual rover.

Figure 3.3: The two highest levels of the functional flow diagram of the rover and its OBCA tasks.

3.1.6. Functional Breakdown

The determination of the requirements for the OBC, leads to the definition of the main functions of the OBC. The OBC is responsible for all CDH processes, which can be divided into three main categories. In the first place there is the management of onboard data. This includes the hazard detection mechanism for the images delivered by the cameras of the rover. Secondly, the OBC is responsible for managing all onboard operations. Finally, the rover needs to manage all data coming in from other rovers or commands coming from the lander or moon base. The functional breakdown structure for the OBC can be found in Figure 3.4.

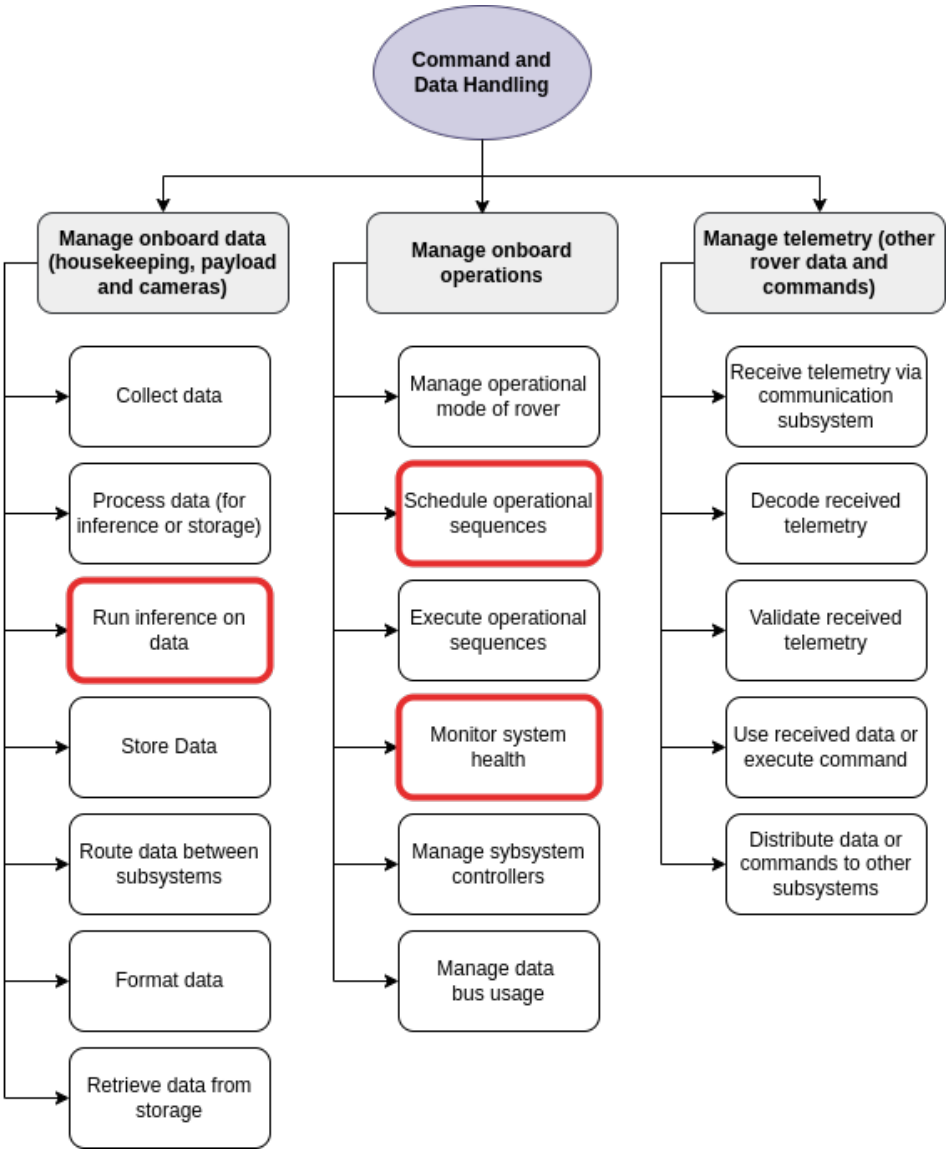


Figure 3.4: The functional breakdown structure for the OBC/CDH subsystem of the next-generation lunar zebro. Inspired by the functional breakdown structure of the first Lunar Zebro mission [26].

In Figure 3.4, the blocks framed in red indicate the operations that will be worked out in further detail. These blocks show operations that could be solved with deep learning processes. Specifically, the deep learning pipeline for rock segmentation will be developed and studied. Moreover, opportunities to leverage the deep learning capabilities of the OBC for mission planning and scheduling will be studied. Finally, opportunities to monitor system health with deep learning-based FDIR methods will be discussed.

3.1.7. Technical Budgets

As part of the requirement engineering process, after the conception of the requirements, the technical budgets that are available for the low-SWaP rover are conceived. For these calculations, a very conservative approach is used, to leave sufficient margin for changes in the rover design in the future and to account for uncertainties.

Energy Budget for the OBC

The available energy for the processing system depends on many factors, among which the design choices made by the Lunar Zebro team. However, estimations of the energy available can be made on the design of the EPS for the first-generation rover [26] as follows:

$$E_{OBC,cycle} = \eta_{OBC} \times N_{batt} \times DoD \times \epsilon \times m_{batt} \tag{3.1}$$

with $E_{OBC,cycle}$ the total energy available for the complete OBC per cycle, η_{OBC} the percentage of the total energy budget available for the OBC in Wh, N_{batt} the number of batteries, DoD the depth of discharge of the battery in percentage, ϵ the gravimetric energy density of the battery in Wh/kg and m_{batt} the mass of one battery in kg.

The solar power generated by the solar panels is stored in four lithium-ion NCR18650B cells [27]. These cells have a gravimetric energy density of 243 Wh/kg, and a mass of 47.5 grams each. Therefore, the total energy capacity of the batteries lies around $(0.0475 \times 243 =)$ 11.543 Wh per battery. Lunar Zebro has five batteries on board (see section 2.1). Moreover, the depth of discharge of the batteries should be considered. For satellites in Earth orbit, the depth of discharge of Lithium Ion cells is limited to about 40% [132], due to the large number of cycles that the satellites have to go through (>10,000 cycles [133]). However, for Lunar Zebro's 14-day mission, the batteries are not expected to go through more than 100 cycles, considering the 5-hour charging and discharging cycles. Therefore, Depth-of-Discharge rates of 80% are acceptable for the Li-ion batteries [134].

Assigning 10% of the energy budget to the onboard computer was determined to be a reasonable initial estimation based on considerations of typical power requirements for similar space missions. This allocation accounts for the substantial energy demands of the rover's locomotion, instrumentation, communication systems, and environmental control while allowing a reasonable share for the computational needs of the onboard computer without excessively straining the overall power resources. This number will serve as an initial budgetary requirement. However, in the sensitivity analysis in subsection 6.2.3, a thorough review and potential adjustment of these allocations will occur.

Using the presented equation and parameter values, a final energy budget of 16621 Joule per 75-minute operational cycle can be found.

Table 3.4: Energy budget per 75-minute operational cycle for the OBCA of the next-generation Lunar Zebro.

Resource	Budget
Energy	16621 [Joule]

Peak Power

The maximum peak power draw for the OBC can be limited in two ways:

1. The power draw of all subsystems shall not excede the maximum power draw that the batteries allow for. The peak power draw of N batteries in series is equal to $N_{batt} \times P_{peak}$. With 5 batteries with a peak power of $36\,\mathrm{W}$ in series, the total peak power of the rover equals $180\,\mathrm{W}$ [27]. With the locomotion being the most power-hungry subsystem, drawing $23\,\mathrm{W}$, this total peak power draw is not considered a limiting factor.
2. The power drawn by all subsystems that require a $5\,\mathrm{V}$ supply shall not excede the maximum power that the $5\,\mathrm{V}$-converter allows to be drawn. The $5\,\mathrm{V}$-converter allows for a peak power of $20\,\mathrm{W}$. It is assumed that the OBC can draw 50% of this for a maximum of 50% of the length of the operational

cycle. This is mainly posed in this way, so that the communication subsystem has the other half of the operational cycle to draw at least 50% of the peak power. Communications for similar purposes require up to $3\,\mathrm{W}$ [135], as well as exemplary miniaturised spectrometer payloads [35], indicating that the set requirements are conservative, even if the most power-heavy subsystems also require $5\,\mathrm{V}$ supply.

In conclusion, the OBC's maximum peak power is limited by the maximum power that the $5\,\mathrm{V}$-converter can deliver. The limit is set at $10\,\mathrm{W}$, which shall not be required for more than 50% of the operational cycle, allowing for enough design space for other subsystems.

Table 3.5: Maximum Peak Power for the OBCA of the next-generation Lunar Zebro.

Resource	Budget
Peak Power	10 [Watt]

Communication Bandwidth

The swarm can only function on the base of continuous messaging between the individual rovers about detected obstacles, positions, locations of high scientific potential and possible paths. To be able to guarantee that these data streams can take place, one should look into the data that a rover can receive. The communication bandwidth of the rover can be limited by two things: the data bus design of the rover and the receiving hardware.

Since the first generation Lunar Zebro rover will be designed to relay data back to Earth directly [26], the design of this communication subsystem can not be taken as an example for this mission. More probably the rover will be equipped with a communication subsystem similar to Perseverance and Ingenuity [136, 137]. The communication system facilitates omnidirectional communication. Its maximum transmission rate is 250 kilobits per second at distances up to one kilometer.

Another consideration is the data transmission design within the rover. The first-generation Lunar Zebro uses RS485 half-duplex connections for the data bus, which can transfer data at speeds up to 1 megabytes per second [26]. The current design includes one direct connection between the communication subsystem and the OBC [138].

Due to constraints imposed by the receiving hardware's data transfer budget, a maximum data rate of 250 kilobits per second will be considered for the reception process.

Table 3.6: Bandwidth for reception of data of the next-generation Lunar Zebro.

Resource	Budget
Bandwidth	250 [kbit/second]

3.2. Design Options

In section 2.4 it was described that there are several categories of devices available for the computing architecture of Lunar Zebro:

- CPU
- GPU
- FPGA
- ASIC & TPU
- VPU

It was also discussed that CPUs are the central part of every computing architecture, as they are required for the general control and operation of the complete computing architecture and the system it is housed in. While GPUs are a collection of CPUs for parallelization purposes, and SoC FPGAs and TPUs include a CPU on a single chip, a VPU like the Myriad X needs to be added to a general computing device. Because the interest of the Lunar Zebro team lies in adding AI capabilities to the rover, the TPU is the only considered ASIC.

Therefore the 5 design options for Lunar Zebro's computing architecture involve:

1. A SBC, incorporating a CPU
2. Single-Board OBC with Embedded GPU
3. Single-Board OBC with SoC FPGA (see subsection 2.4.3)
4. A SBC, with an Additional TPU Board
5. A SBC, with an Additional VPU Board

All devices, except the additional TPU and VPU will incorporate essential components including memory, peripherals, and interfaces. These constructions are all explained in section 2.4, including figures displaying the connection between the accelerators and the CPU. First these devices will be compared, after which the actual components will be discussed in subsection 3.2.2.

3.2.1. Design Option Evaluation and Selection

Following the identification of potential computing architectures for the Lunar Zebro project in section 3.2, this section aims to methodically evaluate and select the most suitable design.

A SBC

The SBC option will be taken into consideration in this research, considering that finding out whether the rover's algorithms and functionalities can run on the simplest architecture and within the design space represented by the requirements, is of great interest. Moreover, its ease of use and general-purpose application make it a critical baseline for comparison.

Single-Board OBC with Embedded GPU

The ease of implementation and high throughput at a relatively low cost [91] are arguments for the use of a GPU SoC. However, the use of these GPUs in space is still in the early stages, necessitating further research and development to adapt them for reliable space operations [139]. Moreover, the power inefficiency [140] compared to other devices such as TPUs [139, 140] and VPU (4 $\times$ less FPS/W) [141] render this design option unlikely to be the best option for low-SWaP space applications.

Single-Board OBC with Embedded FPGA

The possibility to port CNNs for FPGAs was discussed in subsection 2.4.3. Moreover, the flexibility of the FPGA is an advantage [91], especially to a complex system like Lunar Zebro, for which this flexibility allows for the implementation of other algorithms, apart from a CNN for hazard detection. However, the development time and difficulty are very high compared to SBCs or GPUs [91]. This comparison is portrayed in Figure 3.5 [91], in which 4 of the platforms are compared intuitively in terms of development time (t), development difficulty (d), cost (c), flexibility (f), and throughput (p). An Microcontroller

Unit (MCU) is an integrated circuit designed to govern a specific operation. It contains a processor core, memory, and programmable input/output peripherals. When algorithms are strongly optimized for implementation, FPGAs can even outperform VPUs for CNN tasks in terms of efficiency (bits/s/W) [141]. Following requirement **LZ-OBC-SYS-005**, for a team like Lunar Zebro, the knowledge and time investment required for implementing CNNs onto FPGA structures might not be available.

Moreover, the increased efficiency (measured in bits/s/W) of FPGAs comes at the expense of higher power consumption when compared to the Myriad X VPU, as reported by Leon et al. [141]. As discussed in the description of the EPS in section 2.1, it is observed that power constraints are more stringent in other extraterrestrial rover designs, largely due to the prevalent use of batteries in parallel configurations [28, 29]. This poses tighter constraints on the peak power draw, meaning that stricter power requirements should be considered.

Furthermore, in the work by Leon et al. [142], tests were performed on the ZYNQ 7020 FPGA, the same board as the first generation of Lunar Zebro will house. Leon et al. [142] compare this FPGA with the Myriad 2 for CNN ship detection, running a 6-layer network with 132,000 parameters to detect ships on 1024 x 1024 RGB and FP16 images. Comparing this network to the SOTA (rock) segmentation networks in section 2.2, MultiResUNet has 7,262,750 parameters and a much more complex architecture. More lightweight UNet variations for rock segmentation still have 1,939,170 parameters [143].

For the ship segmentation task 'almost all the chip resources' of the ZYNQ 7020 are required for inference [141]. Knowing that the size of the tested CNN is much smaller than the SOTA rock segmentation algorithms, this means that more resources would be required than the ZYNQ 7020 can offer, especially considering requirement **LZ-OBCA-OPER-003**, in which it is stated that resources should be kept available for other operations. Optimizing the Convolutional Neural Network (CNN) implementation to meet specific constraints is deemed beyond the scope of this project.

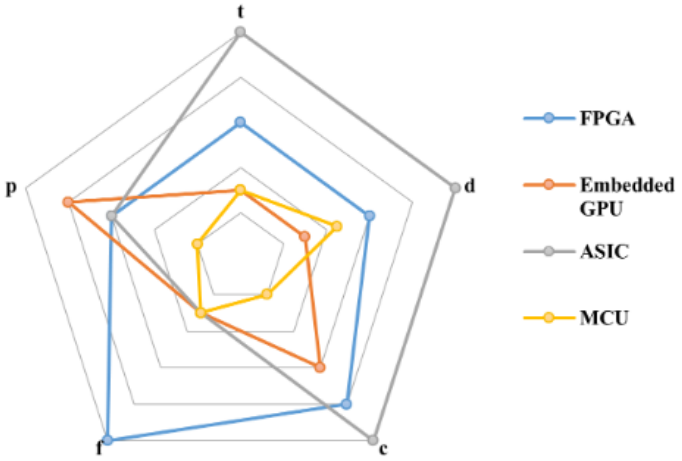


Figure 3.5: Comparison of 4 of the platforms on 5 aspects: development time (t), development difficulty (d), cost (c), flexibility (f), throughput (p) [91]

In conclusion, the higher power usage, resource demands, and notably extensive optimization time present significant drawbacks when considering the FPGA design option. Compared to the XE2 board, these factors render FPGA less favorable or, at the very least, beyond the current scope.

A SBC, with an Additional TPU Board

In the work by Ostrowski et al. [144], the Coral USB, equipped with the EdgeTPU, demonstrates competitive inference times when compared to the Myriad VPU, particularly with the CNN MobileNet. The Coral USB also shows advantages in power efficiency, requiring less energy in both idle and peak

power states [144]. Despite these benefits, a significant concern arises with the Coral TPU regarding its accuracy. The observed loss of accuracy [144] is a critical drawback, primarily attributed to the mandatory 8-bit representation on the Edge TPU [145]. This quantization step by definition leads to a loss of precision and accuracy. While the TensorFlow Lite compatibility facilitates the porting of CNNs to these devices [145], the compromise in accuracy is a substantial issue.

Besides comparing the EdgeTPU on the Coral USB with the Myriad X, at least as important are the designs of the complete boards housing both processors. The Ubotica CogniSat-XE2 [146] is designed to prioritize low-power usage, a critical feature for deployment in energy-constrained environments such as small rovers. In contrast, NASA's SpaceCube Edge TPU SmallSat Card [93], housing three coral Edge TPUs, while offering high inference speed and redundancy through its multiple Edge TPUs, does not explicitly emphasize low-power and low-energy usage to the same extent, making it less ideal for applications where energy availability is a limiting factor.

Due to the accuracy compromise of the Edge TPU and most importantly due to the lower power envelope of the XE2 board, the focus is shifted to the XE2 housing the Myriad X VPU, which provides a balance of power efficiency, inference speed, and maintains the high accuracy standards required for the project's success.

A SBC, with an Additional VPU Board

In the discussions of the SoC FPGA and TPU options, it was shown that the performance of the Myriad X VPU is competitive with those design options. However, besides its specialization in CNNs, which is of special interest to Lunar Zebro anyway, the ease of implementation of CNNs onto the Myriad X is a huge advantage to a team with constrained engineering time. Moreover, Ubotica's interest in supporting the Lunar Zebro mission through help with implementation and possibly financially is another argument to make for the consideration of this hardware for enabling AI processes on board Lunar Zebro.

3.2.2. Considered Hardware

The two considered design options are thus determined to be:

1. A SBC
2. A SBC, with an Additional VPU Board

The hardware evaluation in this thesis initially revolves around two boards: the Xiphos Q7S, which will function as the OBC in the first generation of Lunar Zebro, and the CogniSat XE2 board. These boards are used as a baseline for testing, and to set up initial budgets for the requirements in subsection 3.1.3. Together, these systems allow for a comparative assessment between traditional computing and AI-accelerated processing.

The Xiphos Q7S houses a FPGA, but this will not be considered for the implementation of the DL algorithms, as explained in the foregoing subsection. However, the FPGA on the Q7S does give the Lunar Zebro to implement other algorithms efficiently in the coming years of development. Only the PS of the Zynq FPGA is considered for running the DL algorithms. Both devices are able to run Linux O/S.

3.2.3. Available Memory

Having conceived the design options, the available memory for operations onboard one rover shall be determined. For this, the memory resources of the Q7S are taken as a starting point for the first iteration of the design.

Non-volatile Memory

The Xilinx Xiphos Q7s module has two MicroSD slots, each capable of storing up to $32\,\mathrm{GB}$. To ensure data reliability, data can be saved in two- or threefold (see Figure 4.12). This redundancy strategy guards against potential corruption or loss, bolstering the rover's resilience in the challenging lunar environment and minimizing risks associated with storage failure during its mission.

Following standard protocols for the CogniSat XE2 platform, the non-volatile memory of the XE2 board is left unutilized, except for storing essential firmware required for the operation of the board itself.

Random Access Memory

The Xilinx Xiphos Q7S has two independent Low-Power Double Data Rate (LPDDR) RAM chips, together summing up to $512\,\mathrm{MB}$, taking into account the RAM space required for ECC. Without ECC, 768 MB RAM is available. For the design option including the CogniSat XE2 board, the object detection, FDIR and part of the mission planning task are executed on the XE2 board. Therefore RAM of the Xilinx Xiphos Q7s will not be loaded as heavily.

Flash Memory

There are two QSPI Flash chips on the Xilinx Xiphos Q7s, each housing $128\,\mathrm{MB}$. This type of memory is often used for storing firmware or other data that needs to be persistently stored and accessed quickly.

4

Detailed Design

The detailed design of the OBCA is discussed in this chapter. This entails determining the required extra mass for the design option with AI-accelerator and the cost to bring it to the Moon, which is described in section 4.1. Furthermore, the data bus for the rover is designed in section 4.2, taking into account the flow of data and the requirements stated. Furthermore, the different algorithms on board are worked out in detail in section 4.3, and the relevant specifications of these algorithms for the OBCA design are determined. Finally, the effects of the radiation environment are studied in section 4.4, highlighting the consequences for the OBCA and the possible measures that can be taken for the different design options.

4.1. Mass and Cost Increase of AI-accelerator

For a design option with an extra board such as the design option with AI-accelerator, the size, mass and cost increase of the complete rover should be estimated. For these estimations, the CogniSat XE2 board is considered as the extra board.

Mass

The first generation of Lunar Zebro is designed as compactly as possible, meaning that additional hardware would require a structural extension to the rover's chassis, designed to accommodate the board with a clearance margin of 5 mm on all sides. This extension is constructed from 2 mm thick aluminum. The precise dimensions of the extension, approximately 109 x 108 x 23.5 mm, are determined based on the board's dimensions and the additional clearance. The calculation of the mass of this aluminum extension is calculated by considering the extra required surface area to cover the new hardware (as indicated in red in Figure 4.1) and by utilizing the known density of aluminum, which is approximately 2.7 g/cm³. The result of this calculation indicates an estimated mass for the chassis extension of around 183.5 grams.

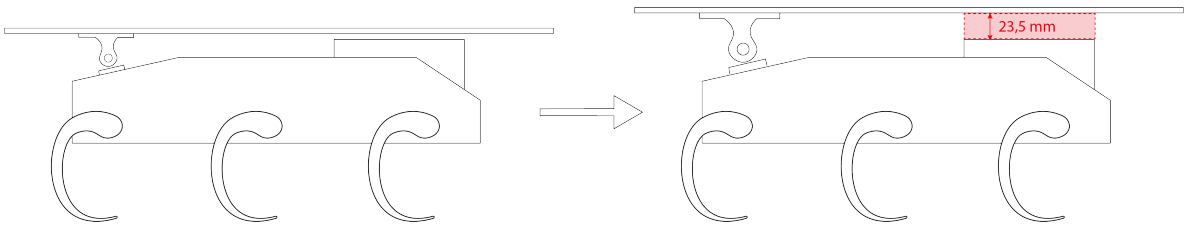


Figure 4.1: A depiction of the required additional volume for the XE2 accelerator.

To secure the XE2 board, four M4 aluminum fasteners are taken into account, which could be attached to the fastening mechanism of the main OBC. These fasteners would add up to a mass of 8 (4×2) grams [147].

Finally, extra cabling would be required; an Ethernet cable and a power cable. Both cables are assumed to be 10 cm in length. These cables add an estimated 30 grams in mass.

In conclusion, including the 80 grams of the board itself [11], the additional AI-accelerator would require a total extra mass of approximately $300\,\mathrm{g}$.

Cost

The maximum extra cost for bringing the AI-accelerator is estimated at $400,000. A small portion of this is attributed to the actual board cost, while the majority is the consequence of the added mass of the board and surrounding chassis material.

Table 4.1: Additional mass and cost required per rover for the design option with the CogniSat XE2

Extra Mass (g)	Extra Cost ($)
300	400,000

4.2. Data Bus

Following requirement **LZ-OBC-INT-011**, all subsystems have to be connected to the main OBC through RS485 communication lines. Therefore, the data bus of the rover is designed with RS485 communication lines between the SBC and every subsystem, except the VPU.

Figure 4.2 illustrates one of the data buses (data bus 1). The SBC serves as the primary communication node, initiating data exchange with three distinct slave units; the PPU, BMS and motor drivers. Each slave device is connected to the SBC via a dedicated 'Select' line, enabling the SBC to individually address each device without signal collision. The data exchange between the SBC and the peripheral devices is asynchronous. This means that each data packet is self-contained with distinct start and stop indicators, allowing the receiver to process data without the need for synchronization to a common clock signal.

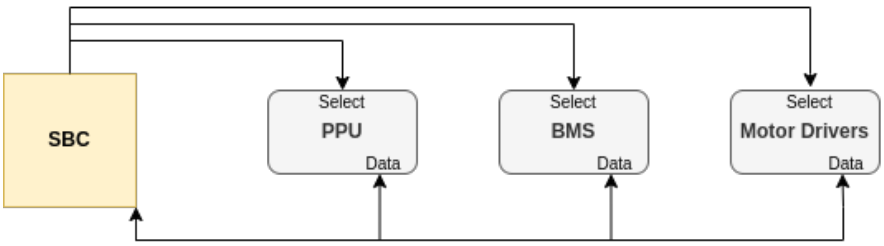


Figure 4.2: A schematic of one of the data busses, following the RS485 data bus convention.

The different data buses are depicted in Table 4.2. Bus 1 to 4 are RS485 data buses, while the red-highlighted fifth data bus is not. This bus will be described hereafter. Firstly, for the design of the data bus, the aim was to have the minimum amount of data buses. This leads to lower mass, required power, system complexity and thus increased reliability. The first data bus is really aimed for smooth motor driver control and powering. This is a continuous process, which should function simultaneously with other processes as well, therefore this needs a separate data bus. The second data bus is for retrieval of data and images from the SHRIMP cameras and payload.
Due to the redundant data bus connection between the SBC and the BMS, the PPU app on the SBC will continue to be in charge of this communication in the event that the PPU fails.

The fourth data bus could be merged with the second but is kept separate to allow for overruling commands from a ground station or human operators on the Moon. Since the static software cycle includes set moments for communication between rovers, this communication bus is only for emergency overrules. A separate connection is put into place between the PPU and Communication subsystem so

that the PPU could command the communications system to send out important data to other swarm members in case of failure of the OBC.

Table 4.2: An oversight of all data buses on board Lunar Zebro. Inspired by [26]

Data bus	Connected Subsystems
1	MCU, PPU, BMS, Motor Drivers of LMS
2	MCU, SHRIMP, payload
3	MCU, PPU
4	MCU, Comms
5	PPU, Comms
6	MCU, VPU

SBC-VPU Board Connection

The main connection between the SBC and CogniSat XE2 can only be either a USB connection or an Ethernet connection. Ethernet is most commonly used in the space industry, according to professionals at Ubotica. Ubotica therefore demanded an Ethernet connection as represented in requirement **LZ-OBCA-SH-019**.
Several secondary connections can be implemented. However, the Q7S board does not have GPIO pins to directly connect these connections to. Therefore, for all the following connections, an external interface connector would be required, introducing more complexity to the system. It is up to Lunar Zebro to decide if the benefits that will be described here outweigh the time investment they will cost.

The following connections should be considered:

1. Controller Area Network (CAN): A CAN interface would allow for communication of the following relevant information and commands:
    - **The average temperature** calculated from the 4 integrated sensors in the XE2.
    - **Self test results** indicating the status of the critical subcomponents.
    - **Board Power**.
    - **Trigger soft reset**.
2. Heartbeat connection through GPIO pins.
3. Direct Latchup interrupt though GPIO pins.
4. Enable lines through GPIO pins. These allow for direct command of a soft reset of the board.

Although it is strongly advised to have a CAN connection to be able to read board power, request temperature and receive self-test results, the OBCA could function without it. The XE2 is self-sufficient, as it will reboot itself when a latch-up or critical SEU is detected. At the very low occurrence rate of SEUs, this can be considered not to be a problem. Moreover, the OBC can infer that a SEU has occurred when a reset takes place. However, for scientific purposes, more detailed data about the cause of the reset might be interesting.

Similar considerations can be made for the other GPIO connections. As the XE2 board is self-sufficient, the OBCA can function without these connections.

As a final note, when none of these connections is available, then soft resets of the XE2 will not be possible. Therefore, PPU will need to cycle the power delivered to the XE2, which is slower than a soft reset. However, as will be discussed in subsection 6.2.2, when the XE2 is taken on board the mission, time will not be a constraint.

In Figure 4.3 all data and power connections to the OBCA structure with XE2 attached are displayed. It is shown that the SBC and XE2 are connected through both Ethernet and CAN. Moreover, the SBC and PPU are connected to four and three RS485 buses, respectively.

In the figure, RS485 transceiver Integrated Circuits (ICs) are indicated by the black diamonds. A transceiver IC for RS485 is a component that integrates both a transmitter and a receiver for serial data into a single module. It is responsible for converting the digital data from the SBC into RS485-compliant electrical signals, and vice versa. Because both devices have built-in RS485 transceivers, they have to be placed externally, but as close as possible to both the SBC and the PPU. The dotted lines in the figure indicate the possibility of including the CAN and GPIO connections discussed above.

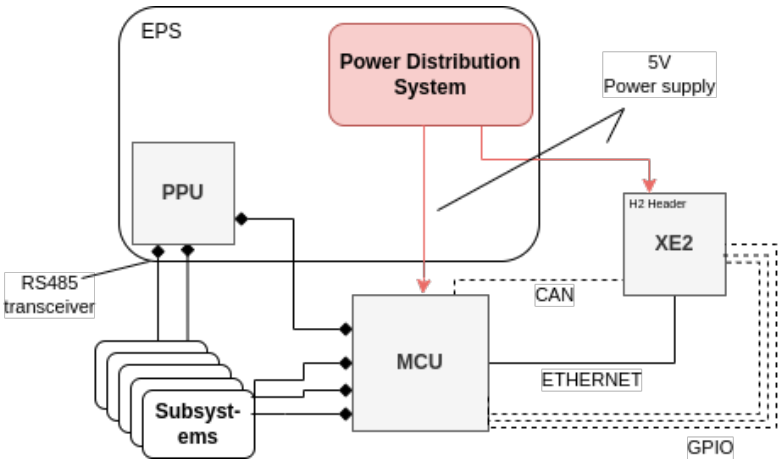


Figure 4.3: Schematic of all data connections and buses & power connections of the SBC and XE2. The black diamonds indicate RS485 transceivers. The dotted line indicates possible connections.

Again, taking the Xiphos Q7s as an exampular SBC, the Xiphos Q7s with all its important features and interfaces are indicated in Figure 4.4 [30]. The power distribution system of Lunar Zebro will be connected to the power input of the Q7s. Moreover, the RS485 transceivers and buses can be connected to the mezzanine connectors.

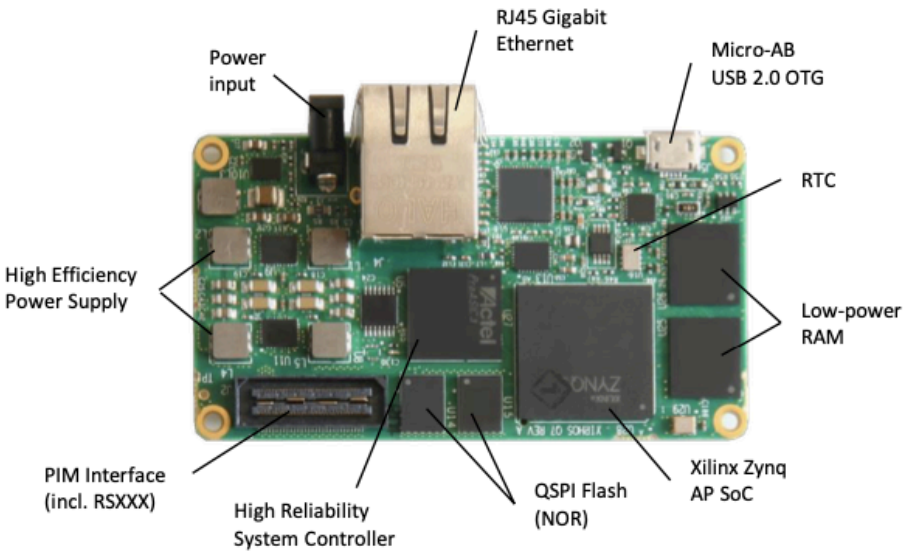


Figure 4.4: The Xiphos Q7s with important features and interfaces indicated [30]

The actual connection interfaces on the XE2 are depicted in Figure 4.5 [148]. The Ethernet connection is marked in green, whilst the CAN interface can be made with pins 1 and 3 of the H2 header. To select an Ethernet connection, Jumper (JMP)4 should not be mounted. Moreover, the JMP1 shall not be mounted for the board to be powered via H1/H2.

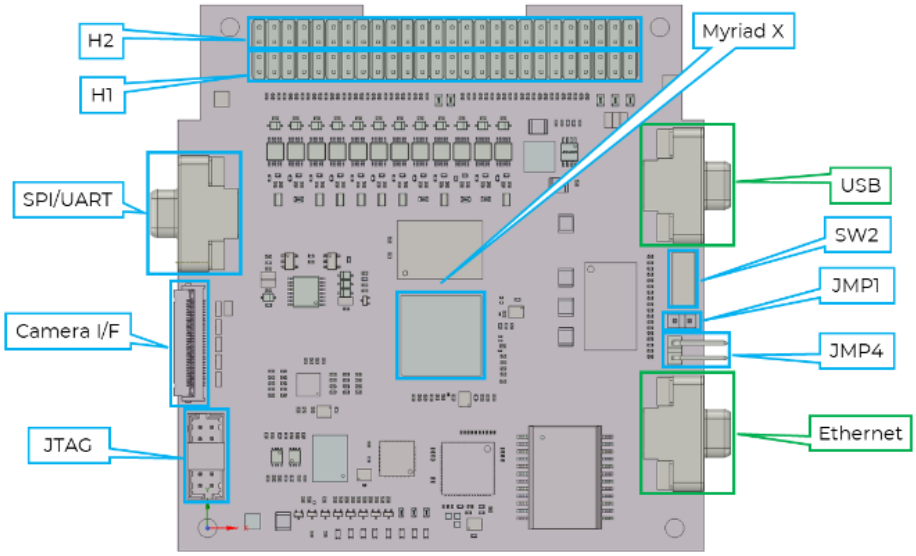


Figure 4.5: XE2 with interfaces and functional blocks highlighted. The two primary data transfer interfaces are marked in green: USB and Ethernet. [148]

The exact connector pins for both CAN and the power connection are indicated by the blue and red boxes in Figure 4.6 [148], respectively. The XE2 has a built-in CAN controller with integrated transceiver [148].

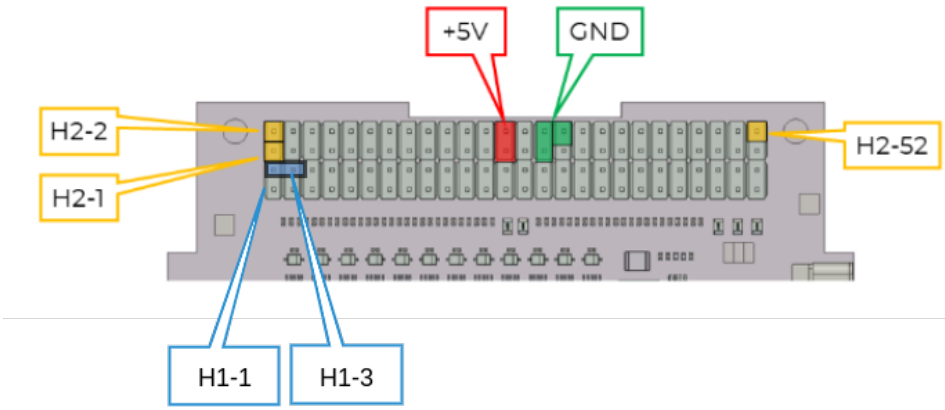


Figure 4.6: The relevant connector pins on the XE2 identified. The power pins are indicated in red and the CAN connector pins in blue. Edited from [148].

4.3. Software Design

In subsection 3.1.5, the operational cycle of the OBC of the rover was discussed. The length of time interval t is an important consideration, as this determines how often the operational cycle needs to take place. Therefore, it is sizing for the required data and energy budgets of the OBC setup. The length of this cycle and the relevant assumptions are discussed in subsection 3.1.4. The different tasks of the operational cycle are outlined in this section.

4.3.1. Nominal Operations

In Figure 2.6 it was shown that different applications will run on the OBC. However, besides these specific applications, the OBC will also need to perform basic tasks, such as task management and distribution of data. These tasks are regarded as 'main processor software' and the power required for these tasks falls under 'typical' or 'nominal' on relevant data sheets [30]. Therefore, the central OBC is

assumed to require $2\,\mathrm{W}$ at all times, for all basic operations.

The required firmware for these operations is assumed to require 10 MB of Flash storage. This is a multiple of the required firmware storage on other missions Ubotica Technologies is involved in, to leave space for unoptimized firmware.

4.3.2. Hazard Detection

Following the research objectives, the specifications of the hazard detection algorithm will be researched and discussed in chapter 5. The exact specifications will be summarized in subsection 6.1.1.

4.3.3. Path Planning

As described in section 2.1, based on earlier research [39, 38] the APF method is assumed for path planning.

- **Duration**: According to the path planning algorithm research for Lunar Zebro by Manteaux [39], a feasible assumption would be to execute the path planning algorithm within $1\,\mathrm{s}$ on the Xilinx Xiphos Q7s. Tests were performed with a Raspberry Pi 4 [149], with a clock speed of 1.5GHz. This is twice as high as the 677 MHz clock speed of the ARM Cortex A9 processors of the Xiphose Q7s, meaning that a delay of at least two times the inference time should be assumed [30]. The time required to recompute a path from one image was measured to be about $0.025\,\mathrm{s}$. Therefore, the duration for this algorithm will be set at a maximum of 1 second. This requirement is thus easily reached by a OBC like the Q7s.
- **Power**: For this period the Xiphos OBC is assumed to have the same power as the rock detection algorithm when executed on the main OBC. This is a conservative estimate, as the path planning algorithm is less computationally complex.
- **Flash**: Similarly, the required Flash memory will be much smaller than that of the convolutional segmentation network. The path planning algorithm is assumed to fit within one-tenth of the required data storage for the segmentation network: $3\,\mathrm{MB}$.

Table 4.3: Required resources for the path planning algorithm.

Criterion	Quantity
Duration	1 s
Power	3.06 W
Flash	3 MByte

4.3.4. FDIR for other subsystems

FDIR can be aimed towards faults in the OBC itself, or in other subsystems. In this section, the focus is on FDIR for other subsystems. On board such a constrained rover, it is not realistic to measure more than the temperature and voltage of a subsystem. Fault detection on the housekeeping data on a nano-rover can be implemented in different ways, in order of increasing complexity:

- **Classic thresholding methods:** predefined upper and lower limits trigger fault handling procedures.
- **Statistical methods:** Utilize statistical analysis to identify deviations from typical operational patterns. This includes techniques like standard deviation analysis for identifying outliers, regression analysis to understand relationships between different operational parameters, and time-series analysis for trend identification.

Statistical methods can help to detect faults sooner, and therefore researching the possibility of adding this feature to Lunar Zebro is of interest. Figure 4.7 [150] depict two different types of anomalies, amplitude and shape anomalies. Shape anomalies are hard to detect with classic methods, but statistical methods can help detect such faults earlier. Murphy et al. [6] discuss the possibilities for AI-based FDIR-related tasks for space applications. Long Short-Term Memory (LSTM) autoencoders were found

to be a useful method for detecting anomalies in spacecraft telemetry. LSTM networks are a type of Recurrent Neural Networks (RNNs). They have feedforward and -back connections, distinguishing this network from regular RNNs. This method definitely needs further improvement from the SOTA, as it also still detects false positives [151]. However, to research the addition of this capability to Lunar Zebro, assuming the use of comparable LSTMs is of interest.

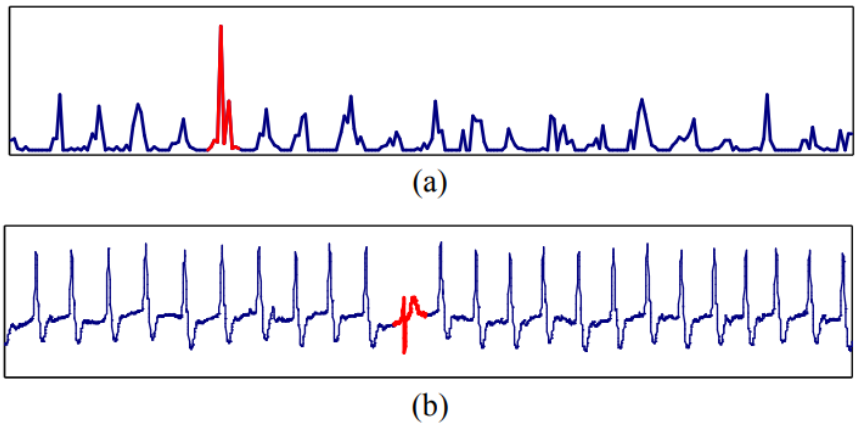


Figure 4.7: Different types of anomalies detecting through time-series analysis: a) anomaly in amplitude, and b) anomaly in shape [150]

A more recent development is the use of Temporal Convolutional Networks (TCNs). This type of network allows for time-series-based analysis with convolutional layers, even for anomaly detection capabilities [152]. This type of network is of special interest when researching the possible use of a VPU like the Myriad X, as this type of network has been run on the Myriad X before. During those tests the Myriad X processed a data rate of 94.96 kbit/s as shared by B. Guesmi of Ubotica Technologies (personal communication, December 2023).

Table 4.4: Size of a housekeeping package for one subsystem

Data	Size [Bytes]
Subsystem ID	4
Status	4
Temperature/Voltage	4
Total	12

- **Non-volatile memory**: To be able to perform deep learning-based FDIR on the housekeeping data, a series of data points shall be saved for the relevant subsystems. In section 2.1 it was discussed that there are 7 main subsystems from which we can expect housekeeping data. In theory, the OBC could receive both temperature and Voltage data from all those subsystems. The first mission shares one form of housekeeping data for some of its subsystems, so this is a conservative estimate.

  Although sharing one data point per subsystem per minute is a conservative estimate as the first mission will only save housekeeping data per subsystem once per 5 minutes, this will be assumed. One data point would consist of subsystem ID (4 bytes), status (4 bytes), temperature (4 bytes) and voltage (4 bytes), all assuming Floating-Point 32 (FP32) accuracy. It is assumed that after three days, the data can be removed from memory. Therefore the required non-volatile storage can be assumed to be:

$$S_{HK} = N_{bytes} \times f \times T_{storage} \times N_{subsystems} = 16 \times \frac{1}{60} \times (3 * 60 * 60 * 24) \times 7 = 483\,\text{kB} \tag{4.1}$$

- **Flash memory**: The network required for deep learning-based FDIR can be assumed to be half the size or smaller than the deep convolutional network required for image segmentation. The required flash storage is therefore assumed to be $14.5\,\mathrm{MB}$.
- **Duration**: For housekeeping data, the size of one data package is summarized in Table 4.4. Assuming this package size, while considering 100 data points per inference, and taking into account the data rate at which the Myriad can process features with a TCN, the inference time on the Myriad can be found to be $\frac{12\times100*8}{94690} = 0.101\,\mathrm{s}$.

  This inference time is about 1/20th of the inference time required for hazard detection on the Myriad X. This ratio is also taken to approximate the inference time for the TCN on the CPU. Approximately the same number is found for the ratio between CNN-based hazard detection on the Myriad X and CNN-based hazard detection on the CPU. The inference time on the CPU for AI-based FDIR will thus be assumed to be $2.8\,\mathrm{s}$ per variable, per subsystem. Although this is a rough estimation, this shows that the order of magnitude will be correct.
- **Power**: Although the duration for inference with the FDIR network is much shorter than for hazard detection, the power consumption during this period will not differ that much. Because of the simultaneous convolutional process of the segmentation network, a slightly higher core utilisation can be expected, but for the purpose of conservative estimation, the power consumption is assumed to be equal to that during inference for hazard detection. ($3.06\,\mathrm{W}$ without XE2 and $4.84\,\mathrm{W}$ with XE2).
- **Inference frequency**: it is unrealistic to assume that the FDIR analysis will need to be run for every subsystem during every operational cycle. Therefore, it is assumed that the analysis is run on two subsystems every operational cycle.

Table 4.5: Required resources for deep learning-based FDIR.

Criterion	Quantity
Non-volatile memory	$483\,\mathrm{kB}$
Duration	$2.8\,\mathrm{s}$ on Raspberry Pi 2 (RP2) and $0.10\,\mathrm{s}$ on the XE2
Flash	$14.5\,\mathrm{MB}$
Power	$3.06\,\mathrm{W}$ without XE2 or $4.84\,\mathrm{W}$ with XE2
Inference Frequency	2 subsystems per cycle

4.3.5. Communication

Given its low-power and space-proven design [135], the ZigBee protocol and communication system are assumed for the intra-swarm communication. In this section, the ZigBee protocol specifications and its implications for the OBCA design are discussed.

The ZigBee protocol is designed upon the IEEE802.15.4-2003 and IEEE802.15.4-2006 MAC protocol [153].

Package Overhead

The data package overhead is of importance because it will shape the quantities of data that need to be transmitted between the swarm. Moreover, this will put requirements on the data bus within the rover as well as the capacity of the OBCA to handle this data.

Several layers contribute to the overhead in each transmitted message [154, 155, 156], as depicted in Figure 4.8 [154]:

- **Preamble Sequence (4 Bytes):** Bits sequence for receiver clock synchronization with the sender, indicating the beginning of the frame.
- **Start of Frame Delimiter (SFD) (1 Byte):** Unique pattern signaling the start of the frame.
- **PHY Header (PHR) (1 Byte):** Contains frame processing information, such as frame length.

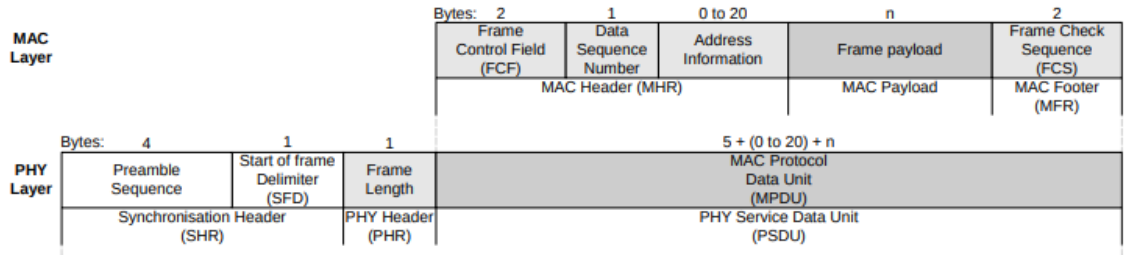


Figure 4.8: Schematic view of the IEEE 802.15.4 packet format [154]

- **Frame Control Field (FCF) (2 Bytes):** Information on MAC frame type, security, acknowledgment, and PAN transmission.
- **Sequence Number (1 Byte):** Tracks frame order and aids in acknowledgment of lost frames.
- **Address Information (0 to 20 Bytes):** Source and destination addresses, variable in length.
- **Frame Payload (n Bytes):** Actual transmitted data, variable in size.
- **Frame Check Sequence (FCS) (2 Bytes):** Checksum for error detection to verify frame integrity.

In total, a worst-case estimate of approximately 31 bytes is required to transmit one message. The IEEE 802.15.4 protocol supports transmission of data packets up to a size of 127 Bytes [157]. Thus, an overhead of approximately 31/127 ≈ 25% should be accounted for.

In section 6.5, the assumed communication workflow within the allotted budgets will be validated.

4.3.6. Operational Modes

Before explaining the power tests performed during the different operations of the RP2 and CogniSat XE2 accelerator, it is important to have a clear definition of the different operational modes of both boards.
The main OBC operates primarily in two modes: *Idle* and *Operational*. During Idle mode, the board remains dormant, consuming minimal power as it awaits tasks or instructions. In contrast, Operational mode engages the board in active task execution, with power consumption varying according to the complexity of the assigned tasks.

Unique to the accelerator is the *Warm-Boot* mode, which offers a distinct advantage in system responsiveness. In this mode, the accelerator operates at a very low-power state while retaining crucial firmware and inference model information within its memory. By circumventing the need for repetitive loading of firmware and model data during startup, the Warm-Boot mode significantly reduces booting time, ensuring swift, yet low-power operation. The main disadvantage of the Warm-Boot mode is that the board will require power constantly, although less than in Idle mode. The exact possibilities and power consumption in power mode are further discussed in Figure 6.1.

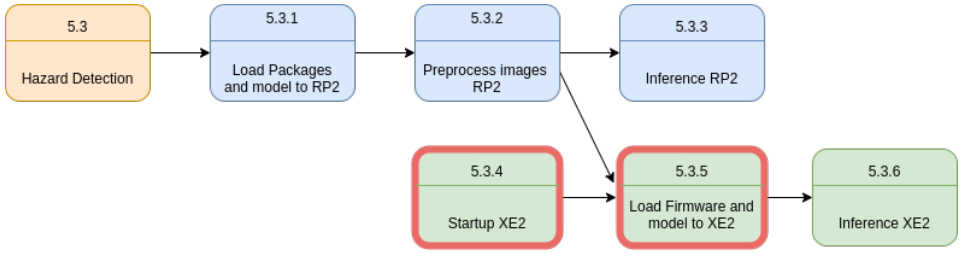


Figure 4.9: The inference pipeline for the setup with only the main OBC/RP2 (blue) and for the setup with the XE2 (green). The red-framed boxes indicate that this part of the pipeline could be bypassed by using warm-boot mode.

4.3.7. Swarming Operations

When designing the OBCA of an individual in a swarm, one has to consider the requirements that stem from a global objective top-down. In other words, one needs to decide what the whole swarm

needs to be capable of and consider the consequences for the design of the individual as is depicted in Figure 4.10. Vice versa, the individual might constrain the possibilities of the global system, thus constraints can be said to flow bottom-up. These requirements and constraints make it so important to go through an iterative design process of the swarm and the individual. The most important considerations regarding those constraints and requirements are the robustness of the individual, the provision of the required tools and sensors, scalability of the swarm, minimal mass and size of the individual robots, centralization considerations, and overarching swarm-level control.

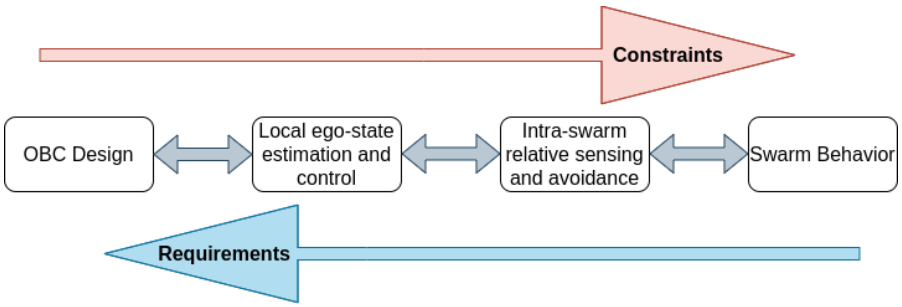


Figure 4.10: Considerations about the swarm lay requirements on the individual and its OBC, where the possibilities for the individual constrain the opportunities for the swarm. Inspired by [158].

Hamann and Wörn [125] provide a detailed mathematical framework for modeling the behavior of robotic swarms. The model is based on concepts from statistical physics and uses the Langevin and Fokker-Planck equations to describe swarm motion.

1. Langevin Equation: This equation models the motion of a single robot, taking into account both deterministic and random components of movement. The deterministic part is influenced by environmental factors, while the random component represents unpredictable fluctuations.

$$\mathbf{R}'(t) = -\mathbf{A}(\mathbf{R}(t), t) + B(\mathbf{R}(t), t)\mathbf{F}(t) \tag{4.2}$$

   Here, $\mathbf{R}(t)$ represents the robot's position at time t, $\mathbf{A}$ is a function representing deterministic motion, B represents random motion, and $\mathbf{F}(t)$ is a random force.
2. Fokker-Planck Equation: This equation extends the model to a macroscopic level, describing the probabilistic behavior of the entire swarm.

$$\frac{\partial \rho(\mathbf{r}, t)}{\partial t} = -\nabla(\mathbf{A}(\mathbf{r}, t)\rho(\mathbf{r}, t)) + \frac{1}{2}Q\nabla^2\left(B^2(\mathbf{r}, t)\rho(\mathbf{r}, t)\right) \tag{4.3}$$

   In this equation, $\rho(\mathbf{r}, t)$ is the probability density of finding a robot at a specific position $\mathbf{r}$ and time t, and Q is a parameter representing displacement due to collisions.

These equations capture the core dynamics of swarm movement, displaying deterministic and random factors. The model can be adapted for various swarm robotics scenarios by modifying the functions A and B based on specific algorithmic strategies and environmental interactions. This framework is useful for predicting and analyzing the collective behavior of robotic swarms, and the complexity of the computations involved is dependent on the specific functions and parameters chosen for A and B.

Function A can be modeled using attraction-repulsion functions [159], that can be obtained from the work of Kim et al. [160]. In this work, the APF path planning concept is extrapolated to function in a robotic swarm. In the swarm, each maneuver minimizes the overall system's artificial potential energy [160]. Simply put; attraction dominates at larger distances to ensure the swarm's cohesion, while repulsion prevents collision at shorter distances. When the swarm is designed and tested, an equilibrium distance shall be carefully chosen.
Furthermore, the total potential U_i^{og} can be calculated as in Equation 4.4, where the potential for obstacle avoidance U_i^o and the potential for group migration U_i^g can simply be added [160].

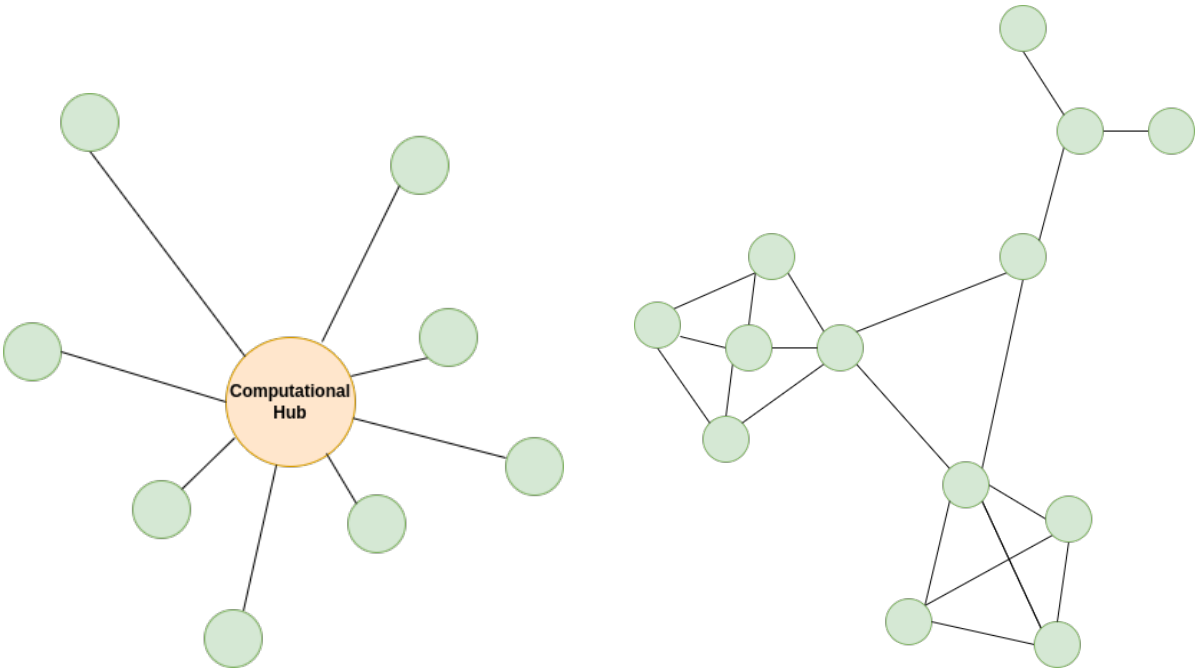
$$U_i^{og} = U_i^o + U_i^g \tag{4.4}$$

Thus, simple attraction and repulsion functions can be used to let the swarm explore coherently, leading to what is called emergent coordination [126]; complex swarm behaviors caused by simple local rules. This makes swarming a relatively small addition to the APF path planning algorithms as designed by Gelling and Manteaux [38, 39]. Both Gelling and Manteaux already discussed the extrapolation of the APF algorithm to a more dynamic environment with moving individuals for Lunar Zebro.

Centralized versus Decentralized Swarm Design

The implementation of the swarm is not set for the next generation of Lunar Zebro. One of the main considerations involves choosing between a centralized [161, 160] or decentralized swarm structure directly impacts the OBC design. A decentralized system, more fitting for lunar exploration due to its adaptability and resilience [162], requires OBCAs capable of independent decision-making and effective local communication [162]. In a decentralized swarm, rovers are designed homogeneously, whereas a centralized system involves different rover, and possibly different OBCA designs, following the requirement flow-down as depicted in Figure 4.10.

There are also many ways for the swarm to keep track of the locations of other individuals in the swarm, which is essential to guard the cohesion and alignment [163] of the swarm. This can be done visually, or with other complex sensors. For Lunar Zebro this will happen through direct intra-swarm communication, meaning that individuals will share their locations with their closest neighbors. This will require the rovers to send out signals periodically, such that other rovers can pick up these locations. Individuals will be able to determine their own locations with a set of Inertial Measurement Units (IMUs) [39].



(a) A centralized swarm **(b)** A decentralized swarm

Figure 4.11: Communication diagrams of a centralized versus decentralized swarm design

4.4. Radiation Risk Assessment

In this section a preliminary radiation analysis is performed for he two design options identified in this work. The devices that need consideration are the ARM Cortex A9 PS of the Zynq 7020 and the Myriad X accelerator. The results of this analysis are expected to provide the necessary background to develop a strategy to ensure the reliability of the computational systems in the lunar radiation environment.

4.4.1. Single Event Upsets

The radiation environment which will be experienced by the Lunar Zebro nano-rover in its 14-day mission on the Moon surface has been modelled by SPace ENVironment Information System (SPENVIS) [164]. All SPENVIS settings are depicted in Appendix B.

In SPENVIS, a lunar surface operation can be best approximated by a 'near-Earth interplanetary' orbit. During solar maxima, the SEU rates increase, as the flux of high-energy particles in the space environment increases. As this radiation analysis should analyze the worst-case scenario, the mission start date is set at a solar maximum; the first of July, 2025 [165]. The distance to the Sun is set at 1 Astronomical Unit (AU). The duration of the mission is set equal to the active period of the OBCA on the Moon; 14 days.

The CREME-96 model [166] is used to simulate solar particles, using its worst week settings. For the solar particle mission fluences, the ESP-PSYCHIC model [167] is used at a 95% confidence level, with no magnetic shielding. GCRs are simulated by the ISO-15390 standard model [168].

Lunar Zebro's chassis is approximated as a 5-sided box of 1.5 mm thick Aluminum. Therefore, the shielded flux is simulated with a total thickness setting of 0.5 g/cm2. Only solar protons are taken into consideration, not trapped protons and trapped electrons, as these exist in the Near-Earth environment.

Finally, the Long-term SEU estimations can be determined in SPENVIS. For this, the shielding thickness is thus set to 0.15 cm. The device material is set to SI (SRIM2008). Finally, SPENVIS requires the shape-sensitive volume of the devices under analysis.

The shape-sensitive volume consists of the X, Y & Z dimensions. The X and Y dimensions can be determined by the SEU cross-section. The sensitive volume in semiconductor devices is influenced by the properties of the silicon wafer primarily because it determines the thickness and material composition of the active region where charge collection and interaction with ionizing particles occur. These properties affect how ionizing particles interact with the material and how charge carriers (electrons and holes) are generated and transported within it. The exact silicon wafer technology is often not shared by manufacturers, as is the case for the Myriad X and ARM Cortex A9 technology.
Therefore, a very conservative estimation is used, following research performed at CERN in 1997 [169]. Since then, semiconductor technology advancements have led to the scaling down of feature sizes in ICs. As technology progresses towards smaller geometries, the physical dimensions of the sensitive volume in transistors and other semiconductor components decrease. Therefore, the assumed shape-sensitive volume of 1 μm^3 is a very conservative assumption, with SOTA chips being produced at sub-micron scale (below 100nm) [170].
This leads to an estimated total of 2.9123E-02 upsets $processor^{-1}$ for the whole mission duration for the Myriad X and the ARM Cortex A9. For now it is thus ignored that the estimated SEU cross-section of the Myriad is much lower than that of the ARM Cortex A9 used on the Q7s board; σ_{Myriad} is 3.95×10^{-10} cm^2/ processor [171].

The ARM Cortex A9 is characterized by a SEU cross-section σ_{ARM_A9} of 6.61×10^{-9} cm^2/ processor [172]. This cross-section is determined by testing the performance of a complete Linux O/S on the CPU and checking for upsets. Hiemstra and Kirsischian [172] also determined the cross-section for a video processing algorithm, but the Linux O/S is determined to be more representative, as this will also be run on Lunar Zebro (requirement **LZ-OBCA-PERF-003**).

To understand what influence the estimated amount of upsets would have on the operation of the OBCA during the mission, one should look into the number of bits per device that are vital for proper operation. In Table 4.6 [173] the division of bits on the Myriad 2 is depicted. The bits in the green rows are considered crucial for success. A bit flip in the registers could immediately misdirect operations or corrupt ongoing processes. Since registers are involved in nearly all CPU operations, their integrity is critical. SEUs in cache memory can corrupt data that the CPU is actively using, leading to incorrect computations or operations. The total amount of critical bits on the Myriad adds up to 5.444 MB (assuming 15 32-bit registers), whilst the total amount of available Bytes counts up to 139.060 MB.

For the ARM Cortex A9, all bits are considered critical for operation, as the cross-section was already determined by looking at upsets during operation of a Linux O/S [172], therefore already ruling out less critical bits.

Table 4.6: The division of memory and cache on the Myriad 2 [173]

Name	Type	Mem/data size
Device level		
DRAM	Memory	134MB
CMX	Memory	2MB
LEON OS L1C	Cache	74 KB
LEON OS L2C	Cache	256 KB
LEON RT L1C	Cache	8 KB
LEON RT L2C	Cache	32 KB
SHAVE L1C	Cache	46 KB
SHAVE L2C	Cache	128 KB
USB bulk transfer	USB PUY	400 KB
Image resize	SHAVE-CMX	614 KB
Data permute	SHAVE-CMX	1.5MB
Inference	Full system	-
Board level		
Flash	Write-read	82 B
PMU	Register read	15regs
SD card	Write-read	1 KB
Reset	Power cycle	-

To account for the division of critical and non-critical bits on the Myriad, the fraction of critical bits can be used to determine the expected SEUs on critical bits, as in Equation 4.5.

$$SEU_{critical} = SEU_{total} \times \frac{N_{bits,critical}}{N_{bits,total}} \tag{4.5}$$

With $SEU_{critical}$ the amount of critical upsets, SEU_{total} the total amount of upsets as determined through SPENVIS, and $\frac{N_{bits,critical}}{N_{bits,total}}$ the fraction of critical bits of the total amount of bits on the Myriad device. This leads to an upset rate of 0.00114 $processor^{-1}$ for the Myriad.

Finally, not all SEUs that occur are as important. As described by Du et al. [174], errors can be classified into three error types:

1. Data error: The observed or computed value was different from the established standard.
2. Program Interrupt: The application software's process was interrupted but subsequently resumed through a software-initiated reset.
3. Time-out (Delay): The testing program failed to generate the anticipated outcome within the allocated time frame.

Du et al. [174] performed soft error experiments on the Xilinx Zynq-7000 SoC, to understand how frequently the different types of errors occur. Table 4.7 [174] shows it was found that 13/19 errors were program interrupts or time-outs, rather than data errors. Data errors are less crucial as flipped bits in the imaging data or collected data are not a direct threat to the operation of the rover. Therefore, 13/19 errors shall be considered critical for mission operation.

During the complete 14-day mission, the Myriad will thus experience 0.001025 upsets, while the ARM cortex will experience 0.02912 upsets per processor. Since there are two ARM Cortex A9 processors on board, 0.05824 upsets can be expected on the total PS of the Zynq 7020.

Table 4.7: Occurences of different error types during radiation tests on Xilinx Zynq-7000 SoC [174]

Blocks	Data errors	Program interrupt	Time-out
Direct Memory Access	6	7	6
Register	0	1	0
Quad-SPI	2	0	0

SEU handling strategy

To accurately determine the fault handling strategy, the upset rates determined in the last section are multiplied by a factor of ten, to make sure that worst-case scenarios are accounted for. In this case, the Myriad will have to cope with about 0.01 upset for the mission duration, and the ARM Cortex PS about 0.5. This is a very rough overestimation, as a safety factor of 10 is applied, and the shielding of the Moon is also not accounted for.

First of all, from the upset rates it can be seen that the resets will not be required many times during the operational period on the Moon. This assures that the rovers' operational cycle will not be broken up very frequently. Consequently, the rovers will not lose each other out of sight due to reboots.

When using a multicore setup such as the dual-core ARM Cortex A9 that the Zynq 7020 houses, roll-back recovery is a useful software hardening technique for tasks for which faults are unacceptable [175]. In this test, each core monitors the other's execution. If an error is detected, the system reverts to a previously saved, error-free state. This technique ensures that even if one core experiences a fault, such as a SEU, the system can recover by using the backup state from the other core [175]. However, if it goes through all the checkpoints and still cannot find a correct, error-free state, the software then needs to take a different recovery action. This might involve resetting the system entirely and restarting the running program, as a last resort to address the fault [175]. For a mission like Lunar Zebro, with its set operational cycles, this complete reset would mean the loss of one operational cycle. At the determined SEU rate for the different devices in the OBCA, this will not lead to large distances between the rovers.

Cui et al. [175] proposed to use Hsiao code [176] to enhance the resistance of on-chip memory against SEUs. This code can correct single-bit errors and detect two-bit errors in the data. Hsiao code does not require data to be saved multiple times. It is a type of error-correcting code designed to enhance memory reliability by detecting and correcting errors in a single set of data. Hsiao code works by adding extra bits (parity bits) to the original data. These added bits help in identifying and correcting single-bit errors and in detecting two-bit errors, without the need to duplicate the entire data set. This makes it a memory-efficient way to increase data integrity, especially in environments prone to radiation-induced errors.

For the computationally heavy CNN- and TCN-operations, rollback recovery is not as suitable, as the algorithms cannot be run simultaneously on a different device than the Myriad. However, it is important to note that these algorithms are reloaded onto the XE2 every cycle, as the firmware and data are saved on non-volatile parts of the Q7S which are less susceptible to SEUs than the volatile parts are [177]. The firmware and data will thus only be present on the Myriad for a short moment, making rollback methods during inference less important. However, methods can be implemented to verify that bitflips have not taken place in storage. For this, algorithms can be stored in 3-fold as depicted in Figure 4.12. Before every upload of the algorithms to the Myriad or CPU, the different copies are compared and integrity checked with methods like MD5 Checksum [178]. Lamorie and Ricci describe how the microSD data is protected even more strongly [179], this confirms the importance of having data stored on this relatively safe non-volatile memory rather than in RAM for longer periods of time.
For these functions, the non-volatile storage on the main OBC shall allow for the storage of multiple copies of the dedicated algorithms.

The short inference time of computationally heavy algorithms on the Myriad X makes the design option

with Myriad more radiation-protected, as the chance that a SEU takes place during inference decreases with shorter inference time.

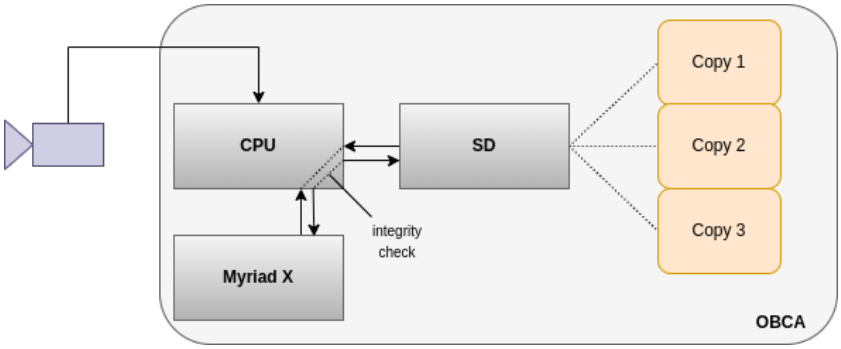


Figure 4.12: Data handling scheme of OBCA including Myriad X

4.4.2. Total Ionising Dose

In the context of the Lunar Zebro mission, understanding the TID and its potential effects on electronic components is vital.

Throughout the SPENVIS simulation, which assumes a start date in July 2025, the radiation environment during a solar maximum is considered. This conservative approach anticipates that the Lunar Zebro mission will not coincide with the peak of a solar maximum, taking into account the Prediction Panel's prediction of Cycle 25 reaching a maximum of 115 occurring in July 2025, with a range of 105-125 and the peak occurring between November 2024 and March 2026. A shielding thickness of 1.5 mm thick Aluminum was taken, again simulating the chassis, which can be approximated by a 5-sided Aluminum box.
For simulating solar particles, the ESP-Prediction of Solar particle Yields for CHaracterizing Integrated Circuits (PSYCHIC) model with a confidence level setting of 95% was employed. For GCRs, the International Organization for Standardization (ISO)-15390 model, an international standard for estimating radiation effects from GCRs, was utilized. This model accounts for fluctuations in GCR particle intensities due to changes in solar activity and the heliospheric magnetic field over 22-year cycles.

The TID received during the operational phase of 14 days is depicted as a function of shielding thickness in Table 4.8. This table provides an overview of the TID expected for different shielding thicknesses during the 14-day lunar mission.
To assess the suitability of electronic components for the mission, it's essential to compare the calculated TID values with the tolerance of the devices. Based on previous testing and data [171], the Movidius Myriad X demonstrated tolerance to a total dose of 14.9 krad (Si) before experiencing failure. This data suggests that the Myriad X's tolerance is suitable for short-duration LEO missions.

Based on the NSREC 2018 study [180], the Zynq7000 chips, which are architecturally similar to the Zynq 7020, exhibited high TID tolerance with minimal parameter degradation (about 5%) and no functional errors at high TID levels. Considering the expected TID of 341 rad with 2 mm aluminum shielding, the Zynq 7020 is likely to withstand the lunar mission's TID, making it a suitable choice for the mission's requirements.

4.4.3. Latch-ups

In the work by Buckley et al., the Myriad 2 was tested with protons in the 30-200 MeV range [173]. No SEL was recorded during any of the tests. Raimalwala et al. report that during proton testing of the Xiphos Q8S up to 105 MeV, 'no destructive latch-up events were detected' [112]. The Q8S is the follow-up board of the Q7S, and Mission Control proposes to use the Q7S for other missions [181, 112], suggesting that the company has also tested the board's latch-up protection for lunar conditions. The protection of the microSD data storage against latch-ups has been elaborately described in the work of Lamorie and Ricci [179]. Therefore requirement **LZ-OBCA-RAMS-004** is met for both design options.

Table 4.8: Dose-depth table for 14 days of operation on the Moon around a solar maximum.

Depth [Al mm]	Dose [rad]
0.05	7766
0.1	4547
0.2	2557
0.3	1775
0.4	1339
0.5	1074
0.6	902
0.8	670
1.0	527
1.5	341
2.0	245
2.5	189
3.0	152
4.0	106
5.0	79
6.0	63
7.0	51
8.0	43
9.0	37
10.0	31

5

Development, Deployment & Resource Evaluation of CNN for Rock Segmentation

In this chapter, the tests for determination of the required resources for DL-based hazard detection are described. In section 5.1, the training of the DL-based segmentation algorithm is discussed, together with the relevant metrics and datasets. Moreover, the conversion of the images and model for inference on the hardware are discussed in this section. In section 5.2, the exact method for the tests is outlined. First of all, the OBC for testing is discussed, after which the test setup and results are discussed.

5.1. Performance Evaluation of Neural Network Inference Across Hardware Architectures

5.1.1. Metrics

Before delving into the specific metrics used to evaluate the performance of the segmentation network, the concepts of True Positives (TP), False Negatives (FN), and False Positives (FP) should be understood. These are depicted in Figure 5.1 [182] and outlined as follows:

- **True Positives (TP)**: True Positives refer to instances where the model correctly identifies pixels as rocks when they indeed are rocks.
- **True Negatives (TN)**: True Negatives are cases where the model accurately identifies non-rock pixels as such.
- **False Negatives (FN)**: False Negatives occur when the model incorrectly classifies actual rock pixels as non-rock.
- **False Positives (FP)**: False Positives represent instances where the model incorrectly classifies non-rock pixels as rocks.

With these definitions in mind, the performance metrics for training and validation of networks can be discussed:

- **Accuracy**: Accuracy quantifies the ratio of correctly predicted pixels, encompassing both rock and non-rock, to the total number of pixels in the dataset. A high accuracy value indicates a higher degree of correctness in pixel classification, providing an essential baseline understanding of the model's performance.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

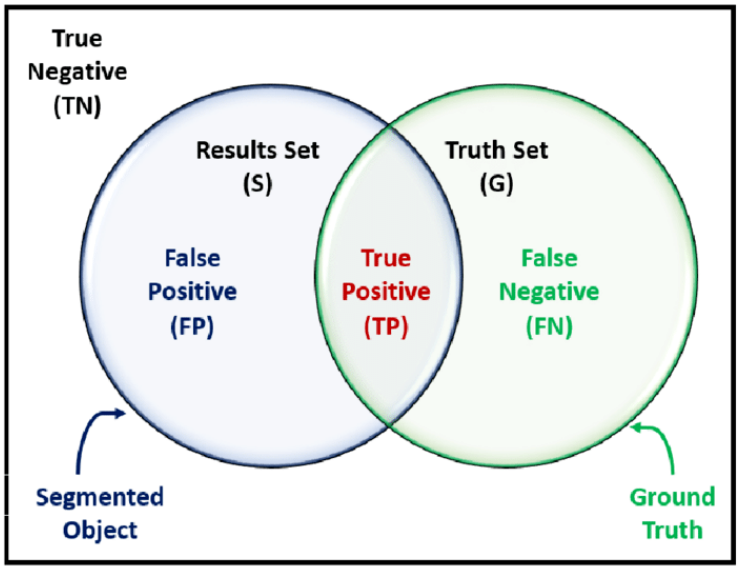


Figure 5.1: A depiction of True Positives, False Negatives, and False Positives in image segmentation [182]

- **Precision**: Precision specifically measures the model's accuracy in predicting rock pixels among all pixels predicted as rock. It delineates the proportion of correctly identified rock pixels from the total pixels labeled as rock. A higher precision value signifies fewer false positives, indicating the network's ability to minimize misclassifications of non-rock pixels as rocks.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5.2}$$

- **Recall**: Recall gauges the segmentation network's ability to capture all actual rock pixels from the entire set of ground truth rock pixels. It calculates the ratio of correctly predicted rock pixels to the total number of rock pixels in the dataset. A higher recall value indicates the model's proficiency in minimizing missed rock pixels, reducing false negatives in the segmentation output.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5.3}$$

- **Intersection over Union (IoU) or Jaccard Index**: The Jaccard Index or Intersection over Union (IoU) metric quantifies the degree of overlap between the predicted segmentation and the ground truth labels, offering insight into the model's accuracy in delineating rock regions. It calculates the ratio of the intersection (correctly predicted rock pixels) to the union (all pixels predicted or labeled as rock), providing a measure of how well the model's predictions align with the actual rock areas.

$$\text{IoU} = \frac{\text{Intersection}}{\text{Union}} = \frac{\text{TP}}{\text{TP} + \text{FN} + \text{FP}} \tag{5.4}$$

- **F1 Score**: The F1 Score is the harmonic mean of precision and recall, providing a balance between these two aspects. It is a more general metric used in various classification tasks, not just limited to object detection or segmentation.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.5}$$

- **Loss:** The loss function is a critical component in training the segmentation network and plays a pivotal role in optimizing the model's parameters. For this study, the binary cross-entropy loss

function was employed to quantify the dissimilarity between the predicted and ground truth segmentation masks. This specific loss function is widely utilized in binary classification tasks, including pixel-wise segmentation, measuring the divergence between the predicted probability distribution and the actual binary ground truth labels. The formula for binary cross-entropy loss is expressed as:

$$\text{Cross Entropy} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \cdot \log\left(\hat{y}_i\right) + \left(1 - y_i\right) \cdot \log\left(1 - \hat{y}_i\right)\right) \tag{5.6}$$

In this formula, N represents the total number of pixels, y_i is the actual ground truth label (binary) for pixel i indicating rock or non-rock, and $\hat{y}_i$ is the predicted probability assigned by the segmentation network for pixel i belonging to the rock class. The binary cross-entropy loss guides the iterative adjustments of the segmentation model's parameters during training, aiming to minimize the error between the predicted probabilities and the true labels.

These evaluation metrics collectively offer a comprehensive framework for assessing the segmentation network's performance in identifying and delineating rocks on lunar surfaces.

5.1.2. The Datasets

In this section, two datasets will be highlighted, as they were either used in foregoing research on DL-based rock segmentation, or used for the inference tests explained later in this chapter. Firstly, MarsData-V2 [183], because it was used to train different UNet configurations to segment Martian rocks by Liu et al. [2]. A comparison of these algorithms is discussed in subsection 5.1.3.
Secondly, the Artifical Lunar Landscape Dataset [184], because of its extensiveness and closer comparison to actual images taken on the Moon. A closer look at both datasets:

- **MarsData-V2**: Developed by a consortium of planetary scientists, MarsData-V2 stands as a comprehensive collection of high-resolution imagery sourced from the Curiosity mission's Mastcam camera between August 2012 and November 2018. This dataset encompasses a rich array of 8390 annotated images portraying Martian terrains, particularly emphasizing rock formations. One sample is shown in Figure 5.2 [183]. The MarsData-V2 has been extensively used to train UNet models successfully for rock identification on Mars [2]. This dataset offers diverse landscapes and annotated rock features, enabling models to generalize well to Martian environments. Only images taken at relatively close shooting distance were used, which can be said to be useful for a rover like Lunar Zebro, which is close to the ground and will therefore not be able to look very far in the distance often. Moreover, given that the Lunar Zebro rover replans its route in each cycle, covering a limited distance, and considering that this planning is impacted by external elements, including interactions with other swarm members, the rover should prioritize the identification of closeby hazards. This can be done by training on a dataset with rocks in the foreground.

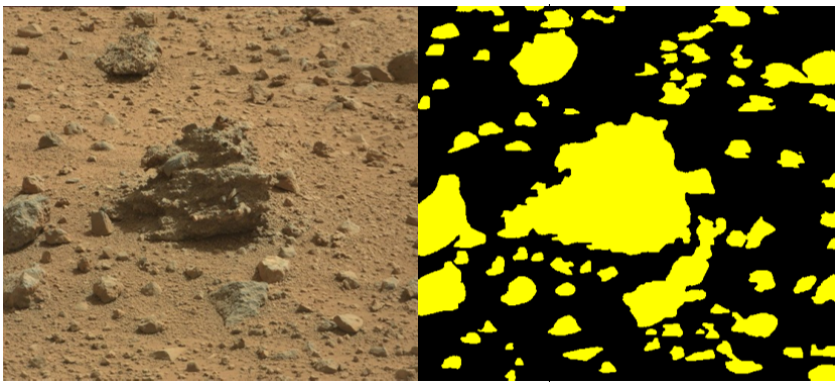


Figure 5.2: An image from the MarsData-V2 dataset, including the ground truth mask [183]

- **Artificial Lunar Landscape**: The Lunar Artificial dataset, on the other hand, is a synthetically augmented dataset made explicitly to simulate lunar surface conditions. The dataset consists of 9766 images. Created by a team of astrophysicists in collaboration with space agencies, this dataset replicates lunar terrain, encompassing varying lighting conditions and surface compositions. It is tailored with synthetic rock formations specifically designed to mimic lunar rock characteristics, allowing for controlled experiments in a simulated lunar environment. As can be seen in Figure 5.3 [184], the masks in the lunar artificial dataset are 4-fold. The sky (red), surface (black), smaller rocks (green), and larger rocks (blue) are separately annotated in the masks.

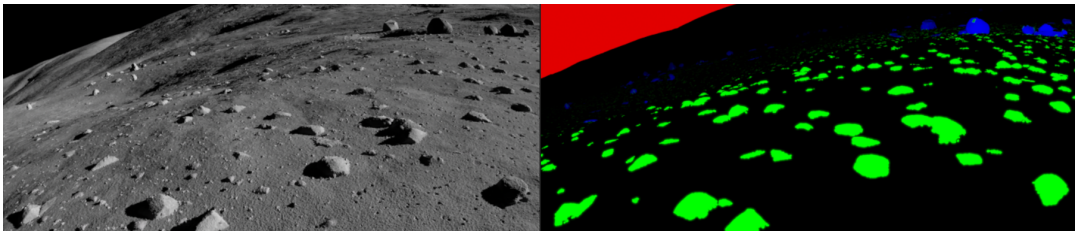


Figure 5.3: An image from the Lunar Artificial Landscape dataset, including the ground truth mask [184]

Both datasets have their advantages and disadvantages. Particularly advantageous to the MarsData-V2 dataset is the dataset's emphasis on close-distance images, mirroring the perspective of a nano-rover, likely to have a limited viewing range. This aspect lends greater realism to the dataset, aligning with the constraints a nano-rover might face in capturing detailed imagery. Moreover, the fact that this dataset consists of real images relieves one's worries about differences between segmented images and real-life images. Furthermore, segmentation success for this dataset has been proven on Martian images, even with MultiResUNet. However, despite its strengths, MarsData-V2 also exhibits limitations, primarily stemming from potential domain shift issues when adapting models trained on Martian landscapes to lunar environments.

In contrast, the Lunar Artificial dataset provides a controlled and synthetic representation of lunar terrains, facilitating systematic experimentation with simulated lunar conditions. This dataset allows researchers to manipulate lighting conditions, surface compositions, and rock formations, providing a controlled environment for testing. Nonetheless, while Lunar Artificial offers control and flexibility, its synthetic nature may limit its ability to perfectly emulate the complexities and nuances of actual lunar landscapes, potentially affecting the model's generalization to real lunar environments.

Due to these advantages and disadvantages experiments were set up with both datasets. The training process is discussed in subsection 5.1.5.

Finally, in computer vision, datasets are usually divided into distinct subsets, namely training, testing, and validation sets. The training set is used to train the model's parameters through iterative learning. The test set, separate from the training data, serves as an unseen dataset to evaluate the model's performance after training. It gauges the model's ability to generalize to new, previously unseen data. The validation set acts as an intermediary subset, aiding in model selection and hyperparameter tuning. By leveraging these distinct subsets, researchers ensure that the model learns from the training data, assesses its performance on unseen test data, and fine-tunes its parameters using validation data, thereby enhancing the model's robustness and generalization capabilities.

5.1.3. Choice of Segmentation Algorithm

In section 2.2, different methods for hazard detection were outlined. Following requirement **LZ-OBCA-FUN-002**, a DL-based segmentation algorithm needs to be chosen. In this section, a comparison of different options will take place, accompanied by a detailed study of the chosen algorithm.

The U-Net architecture, particularly its advanced variant MultiResUNet, has shown remarkable performance in segmentation tasks, crucial for meeting the high accuracy stipulated in **LZ-OBCA-FUN-002**.

MultiResUNet, evolving from the standard U-Net, has been specifically designed to address diverse scales and complexities in images encountered during segmentation tasks. This capability allows it to capture detailed features like fine textures of rocks as well as larger-scale structures such as craters and slopes, essential for effective hazard detection on Martian terrain.

Liu et al. conducted a comprehensive study of algorithms for rock segmentation on the MarsData-V2 dataset [185]. As presented in Table 5.1, MultiResUNet outperformed other U-Net variants in precision, recall, and F1-score, while achieving near-top performance in terms of accuracy on Martian images [186]. Furthermore, it demonstrated high frame rates (FPS). Moreover, it demonstrated fewer false negatives when tested on the SynMars dataset, a dataset featuring many small rocks [186], [185]. This aligns well with requirement **LZ-OBCA-FUN-002** to detect rocks larger than 3 cm, underscoring MultiResUNet's suitability for this application.

Therefore, based on its demonstrated ability to handle the segmentation of complex and varied terrain features effectively, MultiResUNet is selected for further inference analysis. Its proficiency in diverse image scales and contexts, combined with its alignment with **LZ-OBCA-FUN-002**'s high accuracy requirement, makes it an optimal choice for hazard detection tasks in Martian exploration.

Table 5.1: The Quantitative performance and computational complexity on the MarsData-V2 dataset of 6 U-Net-based segmentation networks and the best performance of classic method for the same application [2]

Category	Methods	FPS on MarsData-V2	Quantitative Performance			
			Acc. (%)	Pre. (%)	Rec. (%)	F1 (%)
Best-performing Classic	RKLRR/KLRD	N/A	79.48	66.38	71.68	68.93
UNet-based	U-Net [187]	84.7	87.89	84.43	82.26	83.33
	Furlán et al. [188]	19.9	85.24	83.57	81.83	82.69
	NI-U-Net++ [189]	46.9	88.13	87.98	84.59	86.25
	UNet++ [190]	52.3	90.06	89.03	88.11	88.57
	UNet3+ [191]	42.2	**91.85**	89.77	88.43	89.10
	MultiResUNet [4]	**92.8**	91.74	**90.85**	**92.16**	**91.50**

The success of MultiResUNet in Martian terrain segmentation suggests its potential for lunar exploration, although there are currently no results for its application on the Moon. Its ability to process images at multiple resolutions aligns well with the Moon's varied geological features, like craters and slopes. Despite the lack of specific lunar application data for MultiResUNet, preliminary tests with simpler U-Net architectures have shown promising results for lunar terrain [192]. These tests were performed on the Lunar Artificial dataset, with validation IoU scores reaching up to 94%, demonstrating the adaptability of U-Net architectures to the lunar environment.

Architecturally, the MultiResUNet introduces a novel framework by integrating diverse resolution levels, crucial for detecting objects of different shapes and sizes [4]. Distinguishing itself from U-Net (Figure 5.4 [4]), the MultiResUNet architecture shown in Figure 5.5 [4] employs *MultiRes blocks* comprising multiple parallel convolutional paths operating at different resolutions. These blocks enable the network to capture information across diverse scales simultaneously. Additionally, residual paths or *Res paths* between the MultiRes blocks facilitate information flow across resolutions, allowing the network to learn and combine multi-scale features effectively.
By harnessing multi-resolution pathways, the MultiResUNet adeptly deciphers intricate details while maintaining a holistic view of the image. This versatility is invaluable for tasks requiring precise delineation of structures varying significantly in size or context within an image.

5.1.4. Image Preprocessing

Before training the network, images and masks need to be preprocessed. First of all, the images can be loaded in RGB format, where the masks shall be loaded in grayscale. Then, the images are consequently resized to <h,w>=<480x640> (customizable) pixels, to represent the image size of the SHRIMP

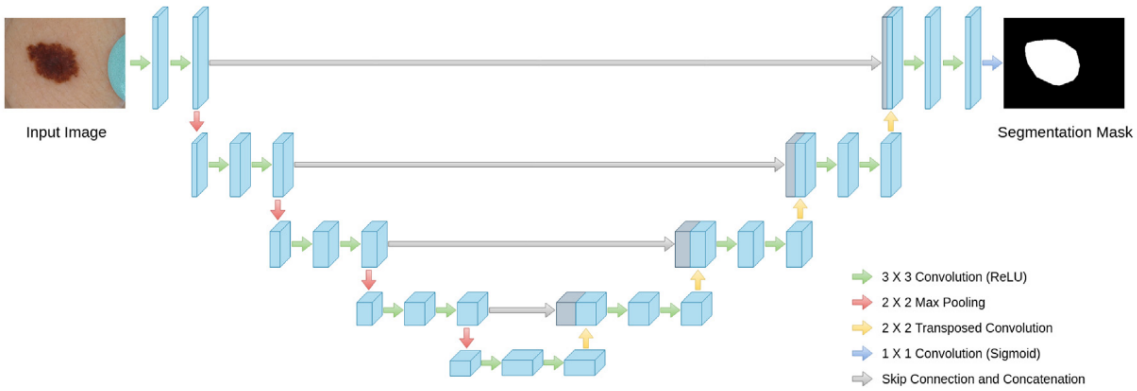


Figure 5.4: The well-known U-Net architecture, characterized by the encoder and decoder pathways, with skip connections between corresponding layers. The different operations between the layers are depicted by colored arrows, explained in the right-bottom corner [4].

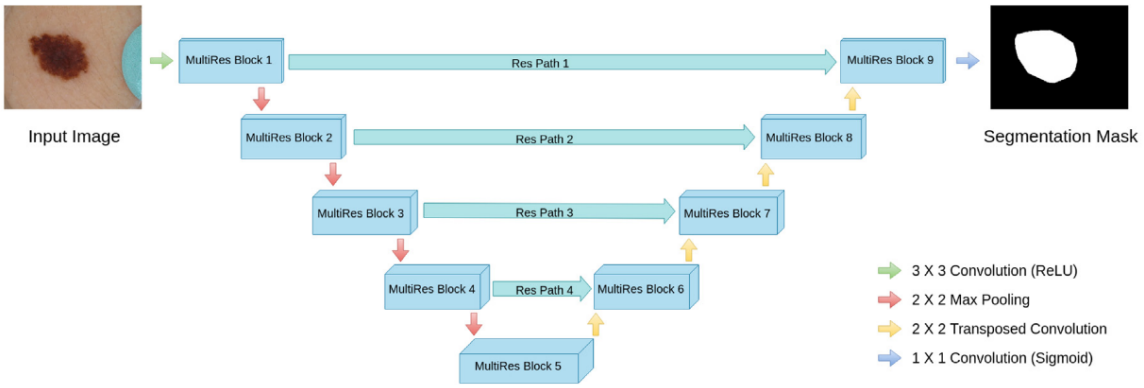


Figure 5.5: The MultiResUnet architecture, defined by MultiRes blocks in the encoder and decoder pathways, linked by so-called Res paths instead of plain skip connections like in UNet [4].

cameras. Bicubic interpolation is used for this. This technique considers the intensities and locations of 16 surrounding pixels (in a 4x4 neighborhood) to calculate the value of a new pixel, resulting in higher image quality compared to simpler methods like bilinear interpolation [193].

Moreover, because the network shall be trained to eventually run inference on the Myriad, it shall be trained with Floating-Point 16 (FP16) images and masks, as this is the data type the Myriad accepts. Finally, the pixel values shall be divided by 255 and the mask values shall be rounded to 0 or 1, leading to black and white masks.

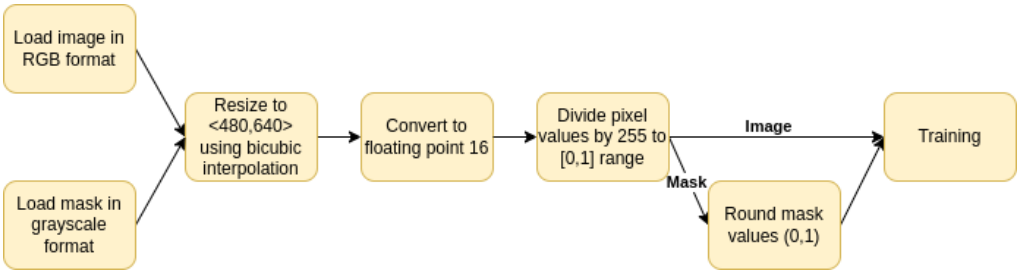


Figure 5.6: Image and mask conversion pipeline for training

5.1.5. Training

To run inference on the different hardware setups, the MultiResUNet model was trained on the MarsData-V2 test set.

As mentioned, training with MarsData-V2 was mostly used in the initial phases of the project, as initially roadblocks were encountered. Because of the implementation requirements of the Myriad X, the deep learning networks that will be run on this chip will need to be able to segment rocks from FP16 images, whilst the network weights are also stored in FP16. Therefore, as an initial step, the effectiveness of MultiResUNet on the MarsData-V2 dataset was tested with the images as well as the network pipeline in FP16 representation. During training runs on a small portion of the MarsData-V2 dataset (100 images), the network quickly learned to identify rocks in the validation set. An initial resulting mask on an image from the validation set is depicted in Figure 5.7. This result shows that MultiResUNet can perform well on these Martian images even though the floating point representation of the images and network is limited to FP16.

For this project, MultiResUNet was trained to match the SOTA results in the work by Liu et al. [194] as in Table 5.1 [186], after this, the results on the Myriad were checked through visual inspection, as past research at Ubotica has shown that final improvements in the inference results of the FP16-trained network can be improved by mixed-precision training. Mixed precision training in deep learning employs both low (16-bit) and high (32-bit) precision numerical formats across various neural network computations. This technique optimizes efficiency by executing more computationally intensive operations with lower precision. Simultaneously, it maintains the necessary precision in areas where accuracy is critical. This approach effectively accelerates training without compromising the quality of the model [195]. The potential application of this method could be explored in further research.

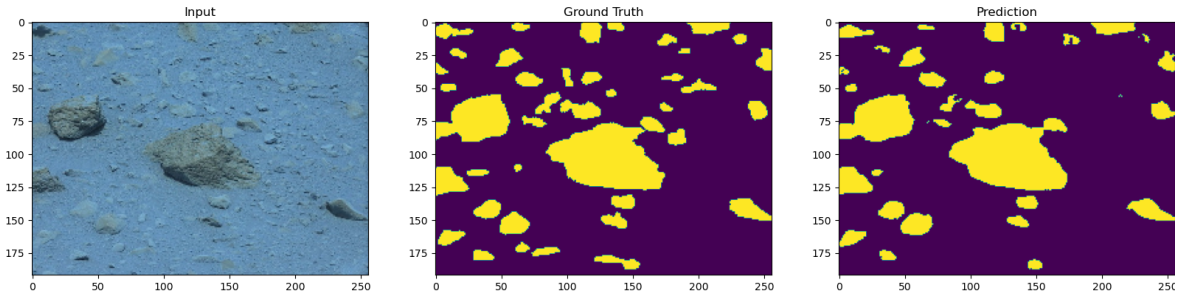


Figure 5.7: Segmentation of an image with Martian rocks with the ground truth mask (middle) and the predicted mask by MultiResUNet (right)

5.1.6. Converting the Model for the Myriad X

When running a model on an ARM Cortex CPU, the CNN can be simply run in tflite format. The trained network cannot be run on the Myriad X hardware directly. The Myriad requires the network to be converted to Ubotica Neural Network (UNN)-format, where the Tensorflow training pipeline saves the model in Hierarchical Data Format version 5 (HDF5)-format.

For this OpenVINO software [196] is used. OpenVINO, an acronym for Open Visual Inference and Neural Network Optimization, is an open-source toolkit developed by Intel. It is primarily used for optimizing and deploying deep learning inference solutions across Intel-based hardware platforms, including CPUs, GPUs, FPGAs, and VPUs like Myriad X. One of its main features is the model optimizer; this tool transforms models from popular frameworks (e.g., TensorFlow, PyTorch, Caffe) into a UNN, optimized for Intel hardware [197, 198].

The step-wise process is depicted in Figure 5.8. This HDF5-formatted network can be transferred to Open Neural Network Exchange (ONNX) format using the $tf2onnx$ function [199]. Next, the model can

be converted from ONNX to UNN with the $convert\_model$ function in python or the mo function on the Command Line Interface (CLI), following steps discussed in the OpenVINO 2022 documentation [198]:

```
mo --input_model <PATH_TO_ONNX_MODEL> --data_type FP16=True
```

'*mo*' stands for *model optimizer*, after which the path to the ONNX model shall be passed on to the converter. One should be aware of the FP16 $--data\_type$ specification. When running a network on the Myriad X, the network needs to be quantized (see Figure 2.4.3), which means that the precision used to represent numerical values within the neural network model during conversion or optimization needs to be changed. This is one of the major reasons that Myriad X can perform its inference so efficiently, but it may also lead to a loss of performance in terms of accuracy.

To execute the above command, the OpenVINO environment shall be used. OpenVINO 2022 is not compatible with every operating system. Therefore, Docker containers can be used to simulate an operating system with the OpenVINO 2022 installation in it [200]. Docker containers are lightweight, standalone, executable packages that include everything needed to run a piece of software, including the code, a runtime environment, libraries, and system tools, isolated from the underlying operating system.

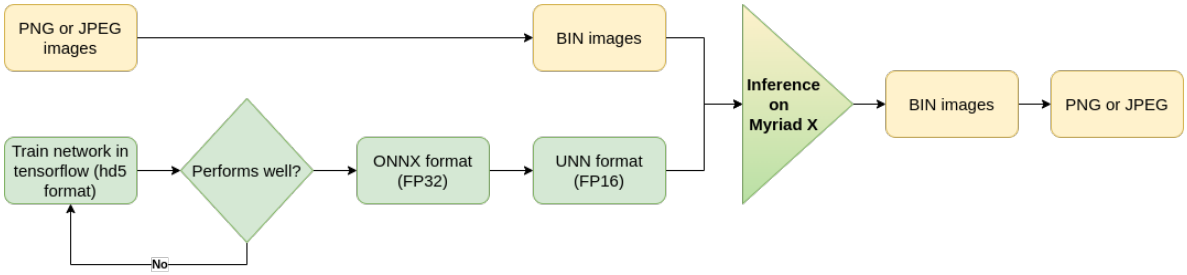
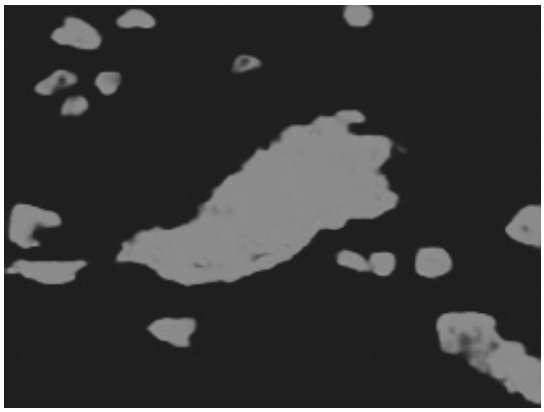


Figure 5.8: Image and network conversion pipeline for inference on Myriad X hardware. The yellow images represent the image pipeline, whereas the green boxes represent the network pipeline.

Besides conversion of the network, the images shall be converted to Binary (BIN) format for Myriad to be able to run inference on them. The written program reads each image and resizes it to the target size of the network. Because MultiResUNet learns features at a specific scale, it might not generalize well to images of different scales. Moreover, the image is converted to a floating-point 16-bit NumPy array, which is then separated into its Red, Green and Blue (RGB) color channels. These channels are interleaved to create a planar data format. The data is transposed to be in a suitable order for further processing, as the network requires the inputs as (Height, Width, Channels) instead of (Channels, Height, Width). Finally, it normalizes the pixel values by dividing by 255 and saves the resulting interleaved FP16 pixel data into a binary file with a .bin extension, effectively converting the image data into a binary format suitable for storage or further analysis.
After running inference on the Myriad X, the resulting masks in BIN-format need to be postprocessed to either Joint Photographic Experts Group (JPEG) or Portable Network Graphics (PNG) format, to allow for visual inspection of the inference results. The inference result obtained is depicted in Figure 5.9.

(a) Another input image for inference on the Neural Compute Stick (NCS) [185]



(b) Final inference result of MultiResUNet inference on the NCS

Figure 5.9: Input image and resulting mask after inference of the converted MultiResUNet on the NCS containing the Myriad X

The importance of maintaining equivalent processing pipelines for both the network model and the images cannot be overstated. This consistency ensures that the network can accurately interpret and analyze the input data, directly impacting the reliability and effectiveness of the inference results. Discrepancies in these pipelines lead to significant performance issues.

5.2. Method

Important considerations for the Power and Energy tests are the devices under consideration, the detailed test setup, and the different test modes.

5.2.1. OBC for Resource Testing

In the process of testing the Xiphos Q7S board, several challenges surfaced within the Petalinux environment, particularly in the cross-compilation of the CogniSatApp application and its embedded packages for this specific development board. Petalinux, akin to Yocto and Linux, serves as a development environment facilitating embedded Linux system creation for various hardware platforms. However, compared to Yocto, Petalinux offers a more streamlined and user-friendly approach, integrating tools for configuring, building, and deploying embedded Linux systems tailored to specific hardware targets. Despite facing hurdles during the cross-compilation for the Q7S board's ARM Cortex A9 cores, it is anticipated that overcoming these challenges in compiling the CogniSatApp within the Petalinux environment for such hardware configurations is plausible with additional time and dedicated resources.

A decision was made to select a more user-friendly board. The primary criterion for choosing this board was the resemblance of it processor to the PS of the Q7S. Emphasis was placed on an ARM-based architecture due to its widespread utilization in modern OBCs and its presence on the Q7S.

The RP2, shown in Figure 5.10 [201] houses an ARM Cortex A7 quad-core setup, which is closely related to the ARM Cortex A9 dual-core structure of the Xiphos Q7S. First of all, they have the same ARMV-7A architecture [202]. The A7 cores are designed with a focus on efficiency [202, 203], while the A9 cores are slightly more complex and therefore will also have a slightly higher power consumption [203].

Overall, the differences between both processor structures will be small in terms of energy consumption and inference time, therefore the RP2 can be used for the verification tests in this chapter, that focus on the use of the PS on both boards. Besides the power consumption of the PS, the power consumption of the rest of the SoCs shall be taken into consideration. Where the RP2 consumes about $1.10\,\mathrm{W}$ in idle state [204] (confirmed in section 6.1), the Xiphos Q7S is expected to have a typical power consumption of $2\,\mathrm{W}$ [30]. Note that 'typical' represents the expected or standard power consumption under normal operating conditions, whereas 'idle' refers to minimal operating conditions.

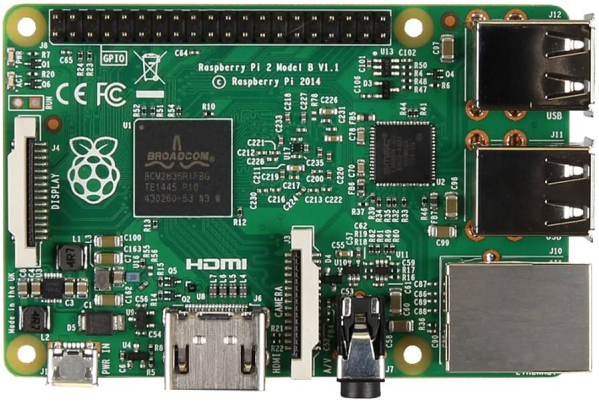


Figure 5.10: RP2 Model B Desktop (with quad-core ARM Cortex A7 CPU at 900MHz) [201]

5.2.2. Test Setup

To measure the power consumption of the OBC configuration for running rock segmentation inference, a test setup was configured. This test setup is schematically depicted in Figure 5.12.

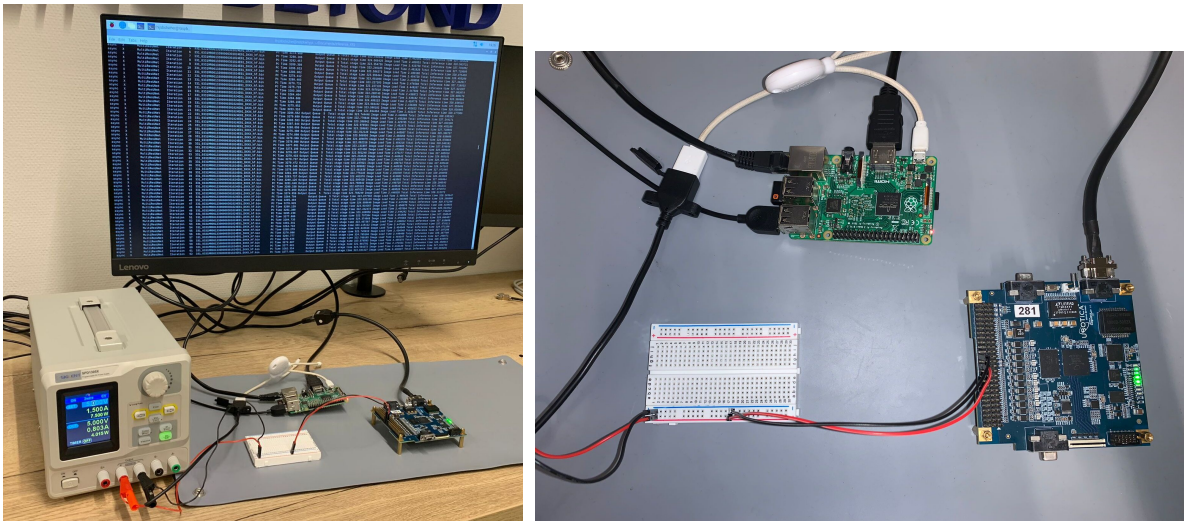
In this setup, a variable power supply (the SPD1305X [205]) was used to deliver a set voltage of $5\,\mathrm{V}$ to both boards. The power supply is connected to a laptop through USB which, through a number of Python scripts, saves and logs Power, Current, and Voltage supplied to the setup over time.

The schematic also depicts the USB connections between the Rasberry Pi 2 and the mouse and keyboard, as well as the HDMI connection with a display. These connections are required to control the Rasberry Pi and view its output. Note that these connections all require power. Therefore, modes were also tested in which these devices were not connected. An oversight of all modes for testing is shown in Table 5.2.

Table 5.2: Modes for power testing

Test Modes	
Board(s)	**State**
Only RP2	No USB devices or display attached
	Nominal
	Inference
RP2 & XE2	Nominal
	Inference
	Low-Power/Warm-Boot

In chapter 4 it was discussed that the XE2 board shall be connected to the main MCU through Ethernet. Therefore, this connection is also used in these tests. For the tests not including the CogniSat XE2 board, this board can simply be disconnected. Pictures of the test setup are shown in Figure 5.11.



(a) Front view of the test setup

(b) Top view of the parallel circuit in the test setup, excluding power supply.

Figure 5.11: The test setup for power measurements of the OBC setups.

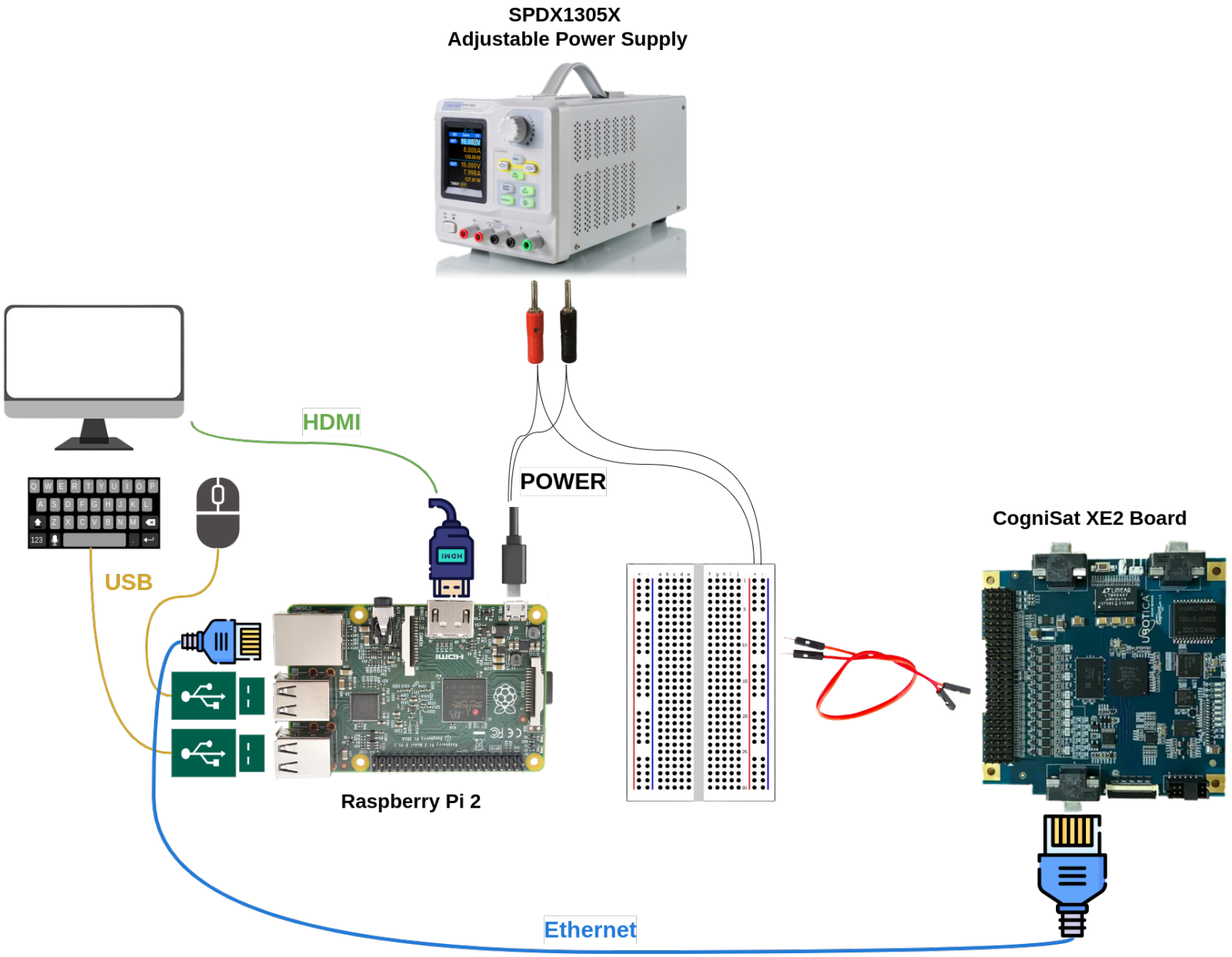


Figure 5.12: Schematic of the test setup for power measurements. The circuit is powered by an adjustable power supply, for which the voltage is kept constant ($5\,\mathrm{V}$) at all times. The Raspberry Pi is connected to a keyboard (USB), mouse (USB) and screen (HDMI). The CogniSat XE2 is powered through the header pins. Communication between the Raspberry Pi 2 and XE2 occurs through Ethernet.

6 Results

In this chapter, the results of the tests are discussed. In section 6.1 the power test results for the different modes are discussed and the specifications of the hazard detection algorithm are then summarized. In section 6.2 the test results are used to simulate the actual duty cycles of the different OBCA designs for the mission lifetime. This shows how the different design options perform within the constricted design space.

6.1. Power Test Results

The outcomes from the conducted tests encompass the mean power consumption of the OBC configurations throughout the preprocessing and inference stages, alongside the duration necessary for executing inference on a specified quantity of images. Furthermore, the RAM demand is of interest, particularly when the inference process is executed on the RP2 itself, as this can be expected to have a heavier workload. The results of the tests are summarised in Table 6.1.

First of all, the nominal power consumption of both design options is measured. That is, the power consumption when no processes are actively run on either device. Note that if the display and keyboard are detached from the RP2, the required power decreases by $0.45\,\mathrm{W}$. The nominal power without devices attached will be closer to reality on Lunar Zebro, thus this will be taken as the baseline power during validation. This means that approximately $0.5\,\mathrm{W}$ can be subtracted from the values in Table 6.1. Furthermore, the results for inference on ten images for both designs are depicted in Figure 6.2 and Figure 6.5. For the run depicted in Figure 6.2, first preprocessing was performed on ten images, after which inference was performed on those ten images subsequently. All numbers for average power consumption and duration per image are depicted in Table 6.1.

Besides the nominal power consumption and preprocessing and inference figures on both devices, the preparation time of both devices is of utmost importance. In Figure 3.3b the pipeline for inference is depicted for both the setup with XE2 (green) and without XE2 (blue). In either setup, the main OBC will be required to load packages for the preprocessing of images, after which it can preprocess the image batch. In the situation without the AI-accelerator, the device is ready for inference.

In the case that the XE2 board is present, it will have to be started up. Afterward, the main OBC will need to send the required firmware, images, and the trained model to the XE2. Only then inference is possible. However, when using the Warm-Boot mode, as discussed in subsection 4.3.6, these steps can be bypassed. These steps are indicated by the red-framed boxes in Figure 6.1.

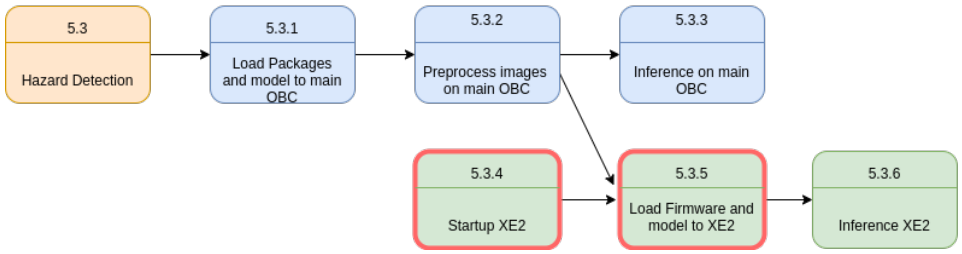


Figure 6.1: The inference pipeline for the setup with only the main OBC/RP2 (blue) and for the setup with the XE2 (green). The red-framed boxes indicate that this part of the pipeline could be bypassed by using warm-boot mode

From Table 6.1 it can be seen that the time required to start up the XE2 and especially to boot firmware and the network onto the XE2 is substantial. Especially when inference is run on a relatively small amount of frames, such as is the case on Lunar Zebro (which travels at low speed and has a low-resolution camera), the startup- and boot time can take the overhand regarding energy consumption. For this reason, the implementation of the warm-boot/low-power mode should be investigated, as this would minimize the required energy for startup and booting.

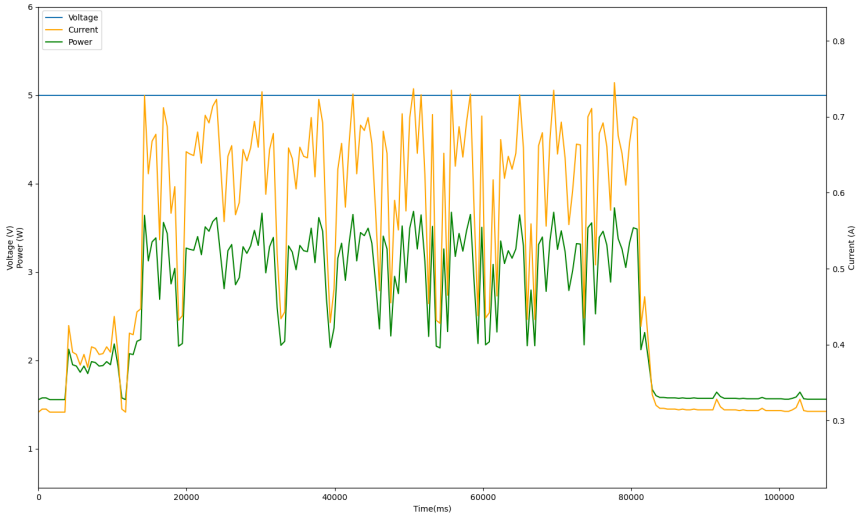


Figure 6.2: Voltage, Current and Power during preprocessing and inference on 10 images on the PS of the RP2. The blue, orange and green lines indicate Voltage, Current and Power, respectively.

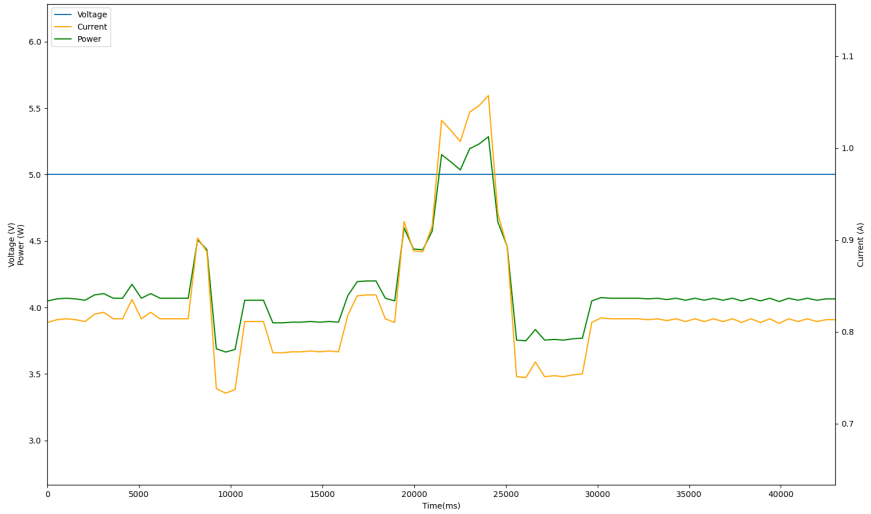


Figure 6.3: Voltage, Current and Power during upload of firmware and model and inference on 10 images on the XE2. The measurements include power consumed by the RP2 and XE2 together. The blue, orange and green lines indicate Voltage, Current and Power, respectively.

Low-power Mode

Power measurements were also done on the low-power mode (see subsection 4.3.6). Initially, it was found that the low-power mode reduces the power consumption from $2.4\,\mathrm{W}$ in nominal condition to $1.8\,\mathrm{W}$ in low-power mode.

In theory, this power usage could be decreased even further. Although Ubotica Technologies has not done research into this yet, there would be the possibility to turn off the Ethernet Controller [206] on the CogniSat XE2 board when no Ethernet action is required. Ubotica's current portfolio consists of satellites in Earth orbit. In these missions, latency is a critical factor. This is usually the reason that the CogniSat XE2 is considered for the mission. If the Ethernet Controller is then turned off for longer periods, the latency increases.

For Lunar Zebro however, the OBC will be able to work in operational cycles of a given length. This would allow for switching on the Ethernet controller at the exact right moment in time. Thus, the Ethernet controller can be turned off during the rest of the cycle. Since the Ethernet controller consumes $0.8\,\mathrm{W}$, the consumption of the XE2 in low-power mode can theoretically be reduced to $1.0\,\mathrm{W}$.
The implementation of the low-power mode is visualized in Figure 6.4. The active modes are alternated by low-power modes, during which only short 'refresh symbols' are transmitted to maintain link integrity [206].

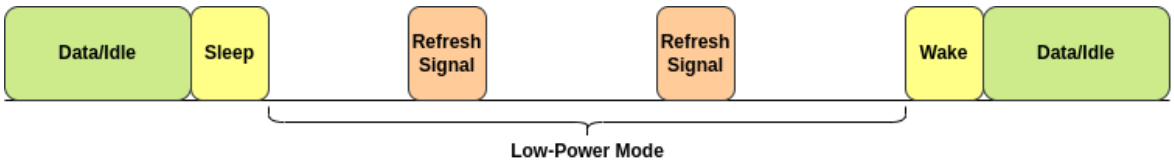


Figure 6.4: The implementation of a Low-Power Mode for the Ethernet controller. Between active moments, only scarce refresh signals are required to maintain link integrity.

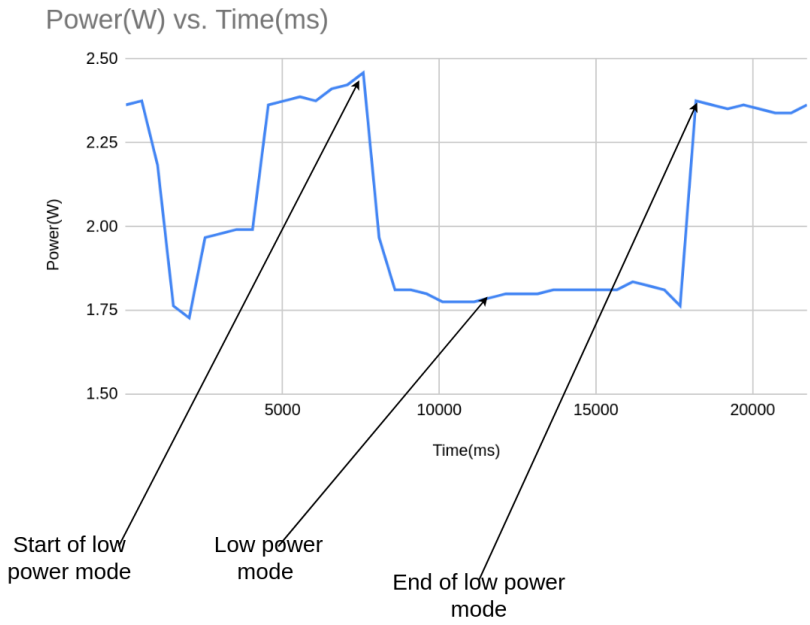


Figure 6.5: Power measurement displaying the difference between nominal and low-power mode of the XE2

Table 6.1: The power consumption by the total OBC setup in nominal state and during all steps in the inference pipeline (Figure 6.1). Note that all values except the first describe the power required when a display and keyboard are attached to the RP2

Scenario	Total Average Power of OBC setup [W]	Time Required [s]
Nominal, only RP2, no display or keyboard attached	1.09	N/A
Nominal, only RP2	1.54	N/A
Nominal, RP2 & XE2	4.04	N/A
Warm-Boot, RP2 & XE2	2.54 [a]	N/A
Startup XE2 (per cycle)	4.04	6.0
Load model to RP2 RAM (per cycle)	3.75	1.95
Load packages, only RP2 (per cycle)	1.95	3.35
Boot firmware and network to XE2 (per cycle) [b]	4.03	12.92
Preprocessing on RP2 (per image)	2.15	0.15
Inference, only RP2 (per image)	3.06	56.22
Inference, RP2 & XE2 (per image)	4.84	2.13

[a] Theoretical value, as deduced in subsection 4.3.6
[b] Tested for several network sizes

The required time to boot firmware and the network to the XE2 is not dependent on the network size. The booting time was measured for several networks with different numbers of parameters and was determined to stay the same. In case Lunar Zebro chooses to use neural networks for several purposes, this is an important consideration.

6.1.1. Hazard Detection Specifications

As a result of the tests, clear-cut specifications can be drawn up on the hazard detection algorithm. This complements the design of the software and its specifications as discussed in section 4.3.
Apart from the values deducted from verification testing, there are two more constraints to consider:

- **Flash memory:** The size of the tensorflow lite version of the rock segmentation model would take up about $29\,\mathrm{MB}$ of Flash storage $14.6\,\mathrm{MB}$ in FP16.
- **Non-volatile memory:** The rover will need to save the images for segmentation. It is assumed that the rover will need to be able to save 10 pairs of images. This is a vast overestimation, as images could be deleted after segmentation. However, this storage space could be useful for storing images of scientifically promising locations. With images from the SHRIMP camera system on board Lunar Zebro containing 640 x 480 pixels [26] and three bands (RGB requires $8 \times 3 = 24$ bits per pixel), the required storage for one image is determined as in subsection 6.1.1. The total required memory for image storage is thus determined at $20\,\mathrm{MB}$.

$$S_{1\_image} = \frac{(640 \times 480)) \times 24}{8} = 921,600\,\text{bytes} \approx 1\,\text{MB}$$

Finally, the specifications of inference with the hazard detection algorithm are summarized in Table 4.3.

Table 6.2: Required resources for the Hazard Detection algorithm.

Criterion	Quantity
Duration	see Table 6.1
Power	see Table 6.1
Flash	$29\,\mathrm{MB}$

6.2. Endurance Test

In Embedded systems design, Endurance tests are often used to evaluate the functioning of a designed system under continuous operation. For this reason, it is important to simulate the constraining factors in the most extreme situations.

In this case, the endurance test aims to verify the requirements on Power and Energy. In the built validation simulation, the different design options are simulated with all software running as designed in section 4.3. All software details follow the assumptions described in this same section. Moreover, the functional flow of the rover follows the design of subsection 3.1.5.

6.2.1. Power cycle

Careful study and simulation of the power cycle has two purposes; first of all, Lunar Zebro requires the peak power drawn to stay under 10 W as described in requirement **LZ-OBC-DES-003**. Secondly, the power cycle can be used to determine the energy consumption of the OBC over time, allowing for a careful analysis of the required energy budget for different OBC designs.
The power consumed by the OBC during one operational cycle is depicted in Figure 6.8. This graph depicts the results as in Table 6.1.
In Figure 6.8, different steps of the operational cycle can be recognised:

I. Loading packages and model to RP2
II. Startup XE2
III. Preprocessing images
IV. Load firmware and model to XE2
V. Run inference on images
VI. Run FDIR on housekeeping data from two subsystems
VII. Path planning
VIII. Only nominal operations

It stands out that the period to startup the XE2 (II) and load firmware and model to the XE2 (IV) is relatively long for the duration of one operational cycle ($120\,\mathrm{s}$), and leads to a high peak power at (III). Therefore, the use of the warm-boot mode should be researched. When making use of this mode, the

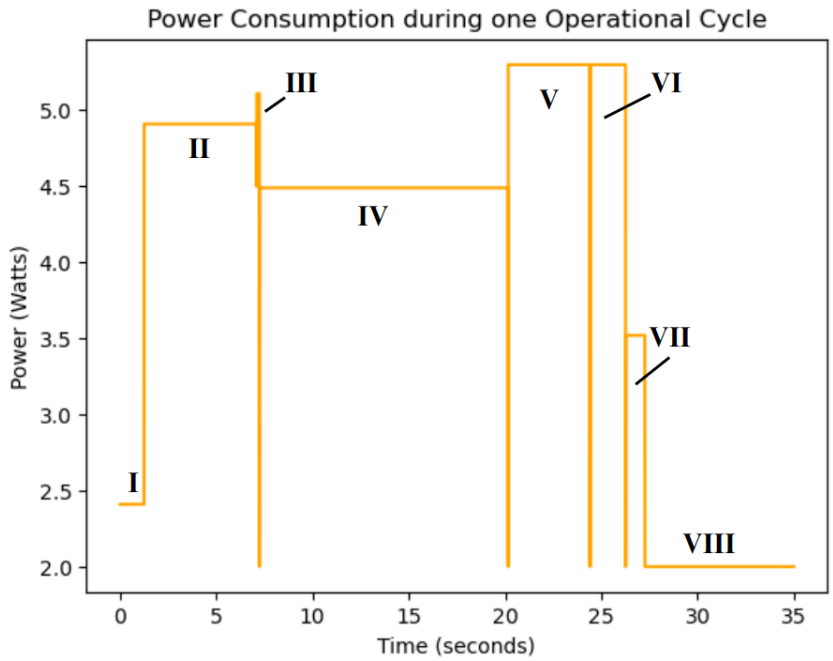


Figure 6.6: Power consumption of OBC including the CogniSat XE2 board during the first $40\,\mathrm{s}$ of the operational cycle of Lunar Zebro, when running hazard detection and inference for 1 rover, assuming stereo vision.

named steps can be skipped, at the expense of a higher nominal power use. Figure 6.7 shows that the period to perform all steps of the operational cycle becomes much shorter when utilizing the warm boot mode. However, setup will require approximately $3\,\mathrm{W}$ in stead of $2\,\mathrm{W}$, typically.

Similarly, the power cycles for the operational cycle for the OBC can be portrayed when the warm-boot mode is utilized. Because of the utilization of the warm-boot mode, starting up the XE2 (step II) and loading the firmware and model to the XE2 (step IV) are not required. Therefore, the duration of the operations all fit within the time span of $15\,\mathrm{s}$. However, this goes at the expense of a higher continuous nominal power, required to keep the XE2 in the low-power warm-boot mode. Where the nominal power without the warm-boot mode is $2.09\,\mathrm{W}$, the nominal power will now be $3\,\mathrm{W}$. Phases of inference for hazard detection (V) and FDIR (VI) have the same peak power requirement as during the experiment without the warm-boot mode.

Figure 6.8 portrays the power consumption of the OBC setup with only the central OBC running inference. It can be seen that steps II and IV are not required for inference on the central OBC. Note the difference in operational time required for all operations; where the operations in Figure 6.8 all take place within $30\,\mathrm{s}$, the operations for the setup without the AI-accelerator takes up almost 4 times as much time. Although the operational time is much longer, this setup deals with much lower (peak) power values than the setup with XE2; where this setup requires a maximum peak power of $3.52\,\mathrm{W}$, the setup with XE2 requires a peak power of $5.3\,\mathrm{W}$.

6.2.2. Energy Consumption

Since power represents the rate of change of energy, the power cycles illustrated in subsection 6.2.1 result in the energy consumption patterns showcased in Figure 6.10. Greater inclines in the energy graph correspond to increased power usage, a noticeable correlation evident upon comparing the two images. The energy depicted on the y-axis is determined from the energy budget in subsection 3.1.7.

From the energy decent per cycle, the energy consumption for the complete operational period can be determined. Figure 6.10 depicts the energy available for the OBC as a function of time for the two different design options. In this depiction, the 75-minute period was taken as operational period between charging periods. As the image portrays an extensive timeframe and is consequently heavily

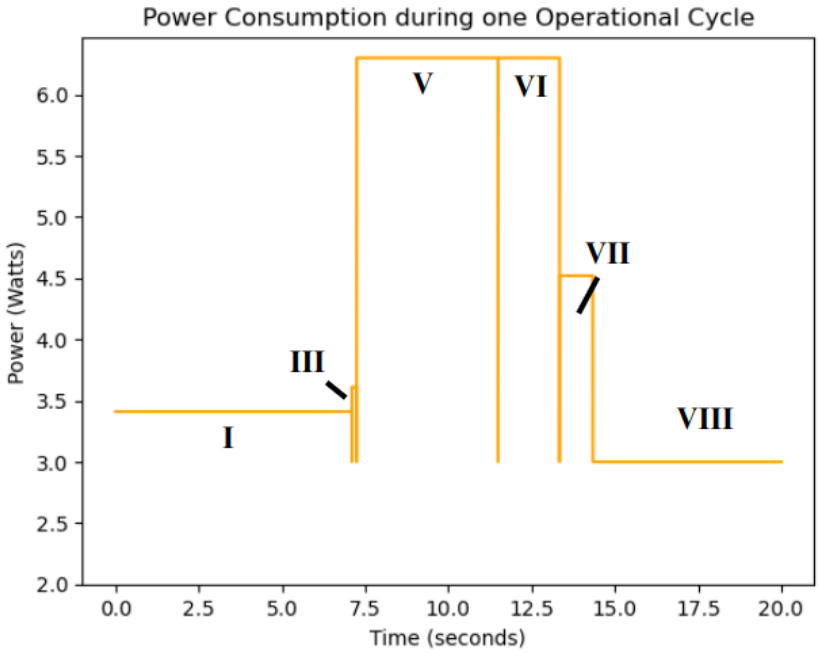
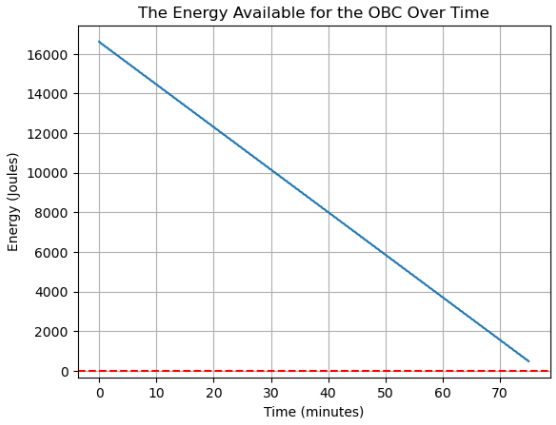
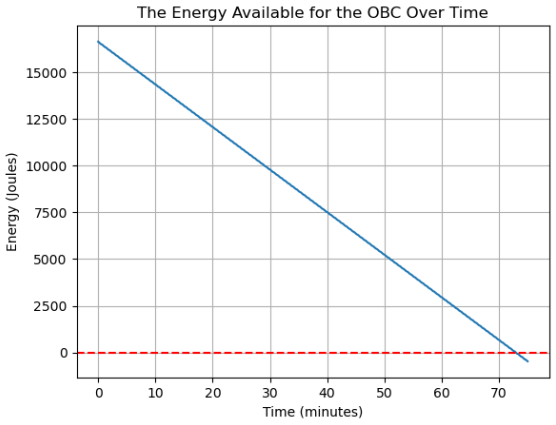


Figure 6.7: Power consumption of OBC including the CogniSat XE2 utilising the warm-boot capability board during the first 20 s of the operational cycle of Lunar Zebro, when running hazard detection and inference for 1 rover, assuming stereo vision.

zoomed out, the fluctuating pattern shown in Figure 6.10 is almost imperceptible, yet on lower scale, this pattern is still present as in Figure 6.9.



(a) The energy available when using a single camera and a OBC setup including the XE2.



(b) The energy available when using two cameras and a OBC setup including the XE2.

Figure 6.10: The energy available for the OBC for a 75-minute period, for an OBC setup with the XE2. The red-dotted line indicates the energy budget limit for the given operational period.

Table 6.3 depicts the energy left after a 75-minute operational period for different design options and operational modes. Besides the different design options for the OBCA, stereo and monocular vision are considered. When no XE2 is used in the OBCA design, the rover cannot perform the required tasks within the 40-second operational cycle (see Table 6.1), making this design option infeasible when using the assumed design space (indicated by 'N/A' in Table 6.3 and Table 6.5).
When the XE2 is used and analysis in two images per operational cycle is required, the rover cannot perform its tasks within the available energy budget. The other three design options are feasible within the given design space. The warmboot mode, as explained in Figure 6.1 leads to a lower total energy consumption.

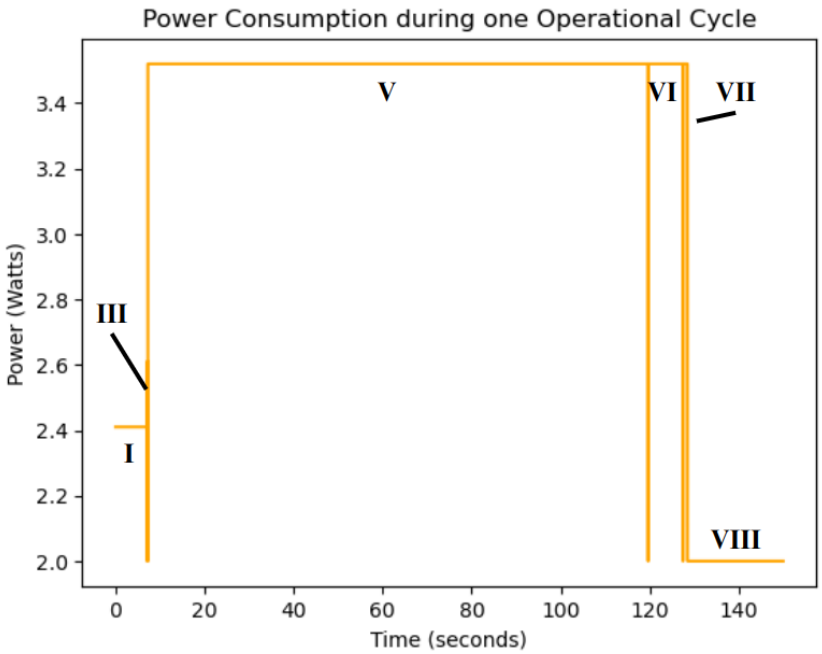


Figure 6.8: Power consumption of OBC setup excluding the CogniSat XE2 board during the complete operational cycle of Lunar Zebro. Inference is run on data from one rover, assuming stereo vision.

Table 6.3: Energy left of energy budget per cycle (16621 J) after 75-minute operational period for different design options and operational modes. The red-marked fields indicate infeasible scenarios.

	Energy Left at End of Operational Phase [Joule]		
	Design Option, Depth Perception Implementation		
Operational Mode	**no XE2, Monocular vision**	**XE2, Monocular vision**	**XE2, Stereo vision**
nominal	N/A	476 (2.9%)	-471 (-2.8%)
warm-boot	N/A	1293 (7.8%)	337 (2.0%)

6.2.3. Sensitivity Analysis

The results as presented in Table 6.3 are based on the assumptions and requirements presented in subsection 3.1.4 and subsection 3.1.3. However, some of these assumptions are subject to change. Therefore, in this section, different parameters will be varied to understand what the consequences would be for the different design options.

Varying Energy Budget

In subsection 3.1.7 the allocated energy budget for the OBCA was determined for one operational cycle. However, the assumptions made for this calculation can possibly change. For example, the addition of 1 battery to the current 5-pack would lead to a 20% increase in the available budget. Therefore, the consequences for the different design options are outlined in Table 6.4.

The four different options depicted in the table all include an XE2, as the inference time of the CPU on one image is longer than the cycle time available.

From Table 6.4 it can be seen that a 10% decrease in energy budget would lead to infeasible energy consumption for all combinations of design options and operational modes. A 10% increase would already allow for the use of the Nominal operation mode, with stereo vision, this can be of importance if the implementation of the low-power mode turns out problematic.

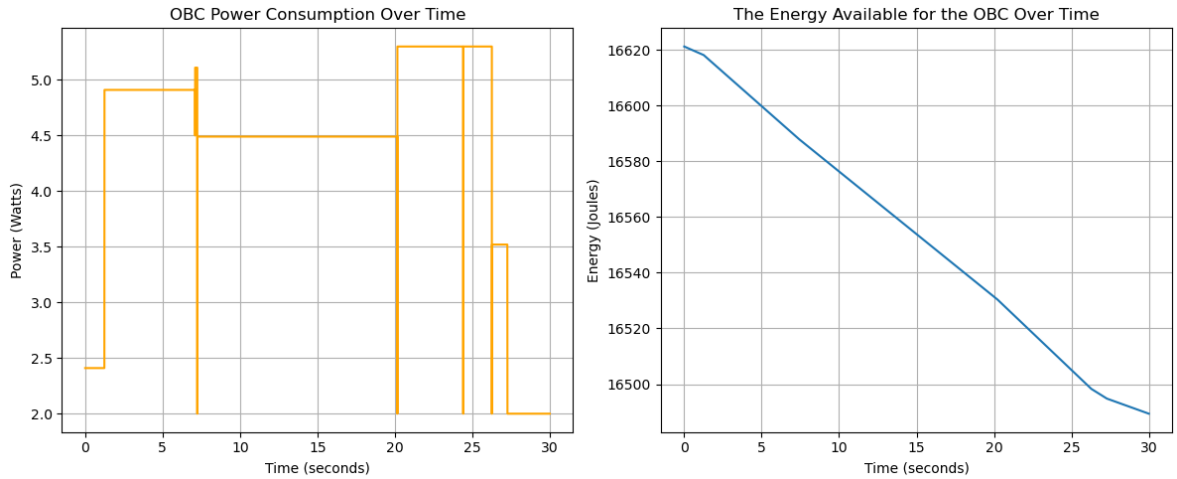


Figure 6.9: Power and energy consumption of the OBC including the CogniSat XE2 board during the first $30\,\mathrm{s}$ of the operational cycle of Lunar Zebro.

Table 6.4: Energy left of energy budget per cycle ($16621\,\mathrm{J}$) after 75-minute operational period for different design options and varying energy budget. All design options consider inclusion of the XE2.

	Energy Left at End of Operational Phase [Joule, % of 16621]			
	Depth Perception Implementation, Operational Mode			
Energy Budget Allocation	**Monocular, Nominal**	**Stereo, Nominal**	**Monocular, Warm-boot**	**Stereo, Warm-boot**
0.8	-2848 (-17.1%)	-3795 (-22.8%)	-2031 (-12.2%)	-2987 (-18.0%)
0.9	-1186 (-7.1%)	-2133 (-12.8%)	-369 (-2.2%)	-1325 (-8.0%)
1.0	477 (2.9%)	-471 (-2.8%)	1293 (7.8%)	337 (2.0%)
1.1	2139 (12.9%)	1192 (7.2%)	2955 (17.8%)	1999 (12.0%)
1.2	3801 (22.9%)	2854 (17.2%)	4617 (27.8%)	3661 (22.0%)

Varying Cycle Time

Currently, the cycle time is based upon the accuracy limit obtained by Manteaux [39]. However, this $0.8\,\mathrm{m}$ can be seen as a lower limit, because Manteaux makes use of an object detection algorithm, rather than more precise segmentation algorithms. Moreover, Manteaux downsamples the images to 160x160 pixels. A higher accuracy limit allows for the rover to cover a larger distance before a new hazard detection cycle needs to be run, resulting in longer cycle times. Therefore in this subsection, the consequences of longer cycle times will be analysed.

As the cycle time increases above $90\,\mathrm{s}$, the CPU would be able to run the inference cycle of Lunar Zebro for monocular vision, and above $150\,\mathrm{s}$ the CPU can fit the operational sequence for stereo vision within the timeframe.

Interestingly, the warm-boot mode can be seen to be advantageous for monocular vision up until cycle times of $40\,\mathrm{s}$. For cycle times of $50\,\mathrm{s}$ and above, the nominal mode requires less energy. This can be explained by the fact that when the cycle time is prolonged, the higher nominal power consumption when the XE2 is not being used becomes predominant.

Table 6.5: Energy left of energy budget per cycle (16621 J) after 75-minute operational period for different design options and varying cycle time. The first two design options are excluding XE2, the final for are including XE2.

	Energy Left at End of Operational Phase [Joule, % of 16621]					
	Depth Perception Implementation, Mode, Design Option					
Cycle Time [s]	**Monocular, N/A, SBC**	**Stereo, N/A, SBC**	**Monocular, Nominal, XE2**	**Stereo, Nominal, XE2**	**Monocular, Warm-boot, XE2**	**Stereo, Warm-boot, XE2**
30.0	N/A	N/A	-1888 (-11.4%)	-3164 (-19.0%)	687 (4.1%)	-589 (-3.5%)
40.0	N/A	N/A	471 (2.8%)	-481 (-2.9%)	1287 (7.7%)	326 (2.0%)
50.0	N/A	N/A	1916 (11.5%)	1150 (6.9%)	1661 (10.0%)	895 (5.4%)
60.0	N/A	N/A	2867 (17.2%)	2229 (13.4%)	1904 (11.4%)	1266 (7.6%)
90.0	2192	N/A	4451 (26.8%)	4026 (24.2%)	2310 (13.9%)	1884 (11.3%)
150.0	4364	1285 (7.7%)	5719 (34.4%)	5464 (32.9%)	2634 (15.8%)	2379 (14.3%)

Centralized Swarm

At higher cycle times, the rover has more available time and energy to process more information per cycle. Therefore, the opportunity arises to run inference for several other rovers. Figure 6.11 depicts the energy decent during one (90-minute) operational cycle of the computational hub of the swarm, when the cycle time is increased to 150 seconds.

As discussed in subsection 4.3.7, the computational hub would run all DL-based tasks; FDIR for other subsystems and hazard detection. It can be seen that even at an operational period time of 90 minutes, the rover can process data for a total of four individuals. At a shorter operational period, the computational hub has even more energy available for inference on data of other individuals.

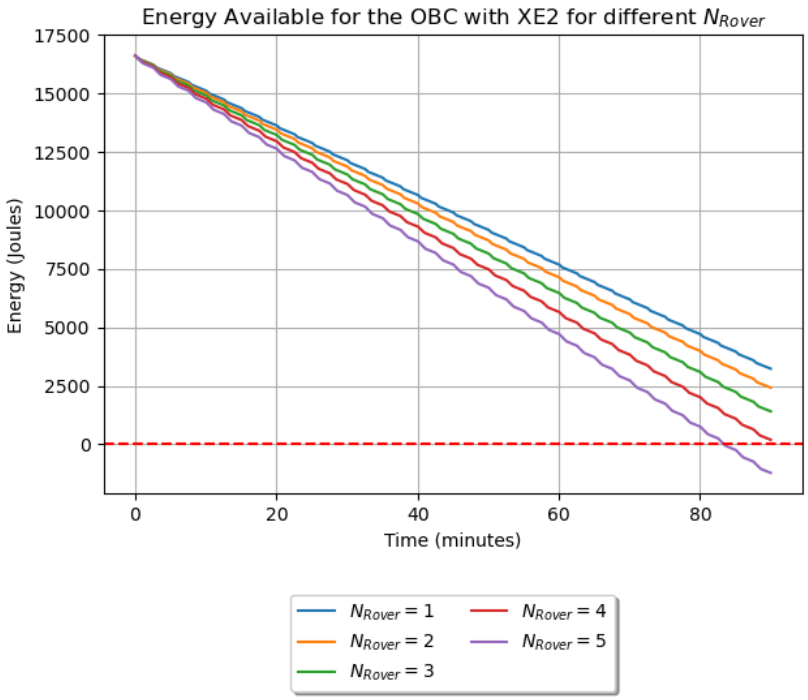


Figure 6.11: The energy consumption during the complete operational period of 90 minutes and a cycle time of 150 seconds for the design option with the XE2, when running inference for 1 to 5 rovers in total. Stereo vision is assumed for this simulation.

The amount of rovers that the computational hub can run inference for is strongly influenced by the

implementation of monocular depth perception. Not only because the rover would require much less energy and time to process the data for one individual, but also because the RAM storage of the rover is limited, which will be further discussed in subsection 6.3.2.
Table 6.6 depicts the energy left of the total OBCA budget of the computational hub, when assuming stereo vision. This energy usage is simulated for different cycle times and amount of rovers the hub runs inference for. At cycle times of 60 seconds and above, the centralized swarm concept becomes feasible. At 120 seconds, the computational hub can run inference for 4 other individuals. The 'N/A' values represent infeasible swarm designs, because the computational hub cannot fit all operations within the cycle time.

Table 6.6: Energy left for the OBCA of the computational hub with XE2 in a centralized swarm, for different cycle times and amount of rovers for which the computational hub runs inference. Stereo vision is assumed.

Energy Left at End of Operational Phase [Joule, % of 16621]					
	Cycle Time [s]				
N_{Rovers}	**50.0**	**60.0**	**90.0**	**120.0**	**150.0**
1	1150.10 (6.92%)	2228.62 (13.41%)	4026.15 (24.22%)	4888.96 (29.41%)	5464.17 (32.88%)
2	-885.97 (-5.33%)	531.89 (3.20%)	2895.00 (17.42%)	4029.28 (24.24%)	4785.48 (28.79%)
3	-3426.94 (-20.62%)	-1585.58 (-9.54%)	1483.35 (8.92%)	2956.43 (17.79%)	3938.49 (23.70%)
4	N/A	-4123.81 (-24.81%)	-208.80 (-1.26%)	1670.40 (10.05%)	2923.20 (17.59%)
5	N/A	N/A	-2181.45 (-13.12%)	189.27 (1.14%)	1739.61 (10.47%)

When monocular depth vision is assumed, a centralized swarm becomes possible at a cycle time of 50 seconds. At 50 seconds, the cycle time is sufficient to perform all processes sequentially.

Table 6.7: Energy left for the OBCA of the computational hub with XE2 in a centralized swarm, for different cycle times and amount of rovers for which the computational hub runs inference. Monocular depth vision is assumed.

Energy Left at End of Operational Phase [Joule, % of 16621]					
	Cycle Time [s]				
N_{Rovers}	**50.0**	**60.0**	**90.0**	**120.0**	**150.0**
1	1915.69 (11.53%)	2866.61 (17.25%)	4451.47 (26.78%)	5212.21 (31.36%)	5719.36 (34.41%)
2	897.65 (5.40%)	2018.24 (12.14%)	3885.90 (23.38%)	4782.37 (28.77%)	5380.02 (32.37%)
3	-372.83 (-2.24%)	959.51 (5.77%)	3180.07 (19.13%)	4245.94 (25.55%)	4956.52 (29.82%)
4	-1895.77 (-11.41%)	-309.61 (-1.86%)	2334.00 (14.04%)	3602.92 (21.68%)	4448.88 (26.77%)
5	-3671.15 (-22.09%)	-1789.09 (-10.76%)	1347.67 (8.11%)	2853.32 (17.17%)	3857.08 (23.21%)

Varying Inference Time

The inference times for the different setups were determined in chapter 5. The results are based on inference runs with MultiResUNet and only for rock detection. The inference time is therefore bound to increase due to several possible factors:

- Higher resolution or larger images.
- Detection of other hazards, such as craters.
- A similar segmentation or classification DL network may be put to the task of determining rocks or regions of more scientific value.

The input of the network can be determining for the inference time, depending on the preprocessing tasks that are performed before inference. As discussed in section 2.1, Lunar Zebro's current SHRIMP cameras take 640x480-pixel images. This input image size might increase drastically if the cameras are also used for other tasks, such as detecting locations of scientific interest. Within Ubotica Technologies experiments have been carried out to find the relationship between the image size and the inference time. It was found that there is an approximate linear relationship between the image size and inference time. If the amount of pixels doubles, the inference time of the Myriad X doubles.

As discussed, MultiResUnet's semantic qualities make it a very adaptable network. It could be used to detect other hazards, such as inclinations, unsurpassable edges and craters. Possibly this would require higher resolution images, which would increase the inference time. Moreover, one should account for additional model complexity when other hazards are segmented by the same network.

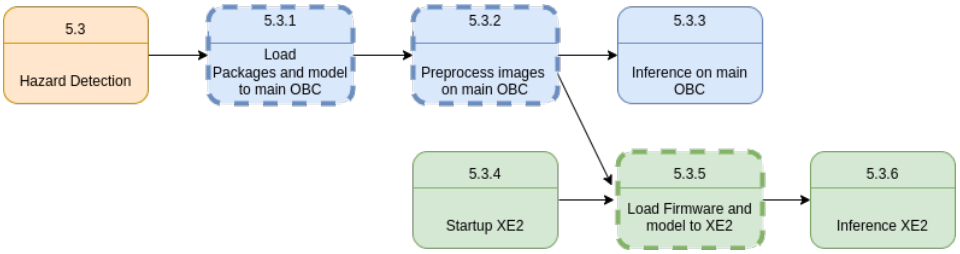


Figure 6.12: The inference pipeline for the setup with only the main OBC/RP2 (blue) and for the setup with the XE2 (green). The boxes with the dashed frames indicate the processes that would need to be (partly) iterated in the case of a second network that uses the SHRIMP images for the determination of regions of scientific interest.

If a DL-based solution would be used on-board Lunar Zebro to recognize rocks or regions of scientific interest, it is possible that images need different preprocessing operations than needed for the hazard detection algorithm. Also, because this would be a whole new network, the network should also be loaded to the central OBC and later to the XE2. Therefore, for this purpose, the relevant operations, highlighted in Figure 6.12 by the dashed frames, are repeated for the new network.

Table 6.8 depicts the energy left for the different scenarios, assuming stereo vision. It can be seen that at longer cycle times, scientific inference, or inference on larger images becomes possible. Above a cycle time of 90 seconds, scientific inference and a 2x multiplication of the image size becomes possible.

Table 6.8: Energy left for different operational scenarios: larger images, extra scientific inference, and a combination of both. All scenarios are for stereo vision.

	Energy Left at End of Operational Phase [Joule, % of 16621]		
Cycle Time [s]	**Inference x2**	**Scientific Inference**	**Combined**
50.0	-364.60 (-2.19%)	-3276.44 (-19.71%)	-4676.23 (-28.13%)
60.0	966.37 (5.81%)	-1460.17 (-8.79%)	-2722.42 (-16.38%)
70.0	1892.95 (11.39%)	-177.13 (-1.07%)	-1254.25 (-7.55%)
80.0	2602.80 (15.66%)	791.55 (4.76%)	-150.93 (-0.91%)
90.0	3184.65 (19.16%)	1566.96 (9.43%)	725.46 (4.36%)

Although it seems likely that the full resolution of the SHRIMP cameras would be used for optimal performance, the input images can also be decreased in size before inference. An example of such a decreased image size is the input data size for a UNet with a pre-trained VGG-16 backbone, which has proven to perform well on the Artificial Lunar Dataset, as discussed in subsection 5.1.5. This proven input image size is 480x480 pixels; a 1.33 decrease in input image size.

The variations in Table 6.8 can of course also be applied to a design with monocular depth estimation implemented. In this case, the cycle time would need to be increased to 50 seconds to be able to run an extra CNN for scientific inference.

6.2.4. Conclusions

In conclusion, the peak power draw is not a concern for Lunar Zebro with the considered design options. A maximum power draw of 5.3 W (« 10 W) is observed. For low-SWaP systems with batteries in parallel, the power envelope might be much more constrained. This might require the use of one single low-power board. Furthermore, the energy use of Lunar Zebro's OBCA is studied for different design options, operational modes and depth perception implementations. It was found that only with

the XE2, the energy budget is sufficient (both monocular and stereo vision), and that the warm-boot mode is advised.

As a base for this research, the operational assumptions described in subsection 3.1.4 were used. When changing these assumptions, some notable conclusions can be drawn:

- **Varying the energy budget:** When the energy budget is decreased, no design options are possible when we assume a cycle time of 40 seconds.
- **Varying the cycle time:** Only at cycle times above about 90 seconds (1.8 m), the CPU is an available design option for monocular depth perception, and only towards 150 s (3m » 0.8 m) for stereo vision. For the design options with the XE2, the warmboot mode shall be used up to and including 40 seconds for both monocular and stereo. At higher cycle times, the nominal operational mode shall be implemented. At cycle times above 50 seconds, the OBCA requires much less energy than the budget allows for. This allows for increasing the 75-minute operational time.
- **Centralized swarm:** A computational hub can only function within the energy budget at higher cycle times. Above 90 seconds, one rover can run inference for 3 individuals, and above 120 seconds even for 5, assuming stereo vision. At 90 seconds the rover can run inference for 5 individuals already when assuming monocular vision.
- **Varying inference time:** Larger images make inference a larger expense, but at cycle times above 60 seconds (1.2 meters), the images can be doubled for stereo vision. Running a second CNN for scientific inference is a more costly endeavor, but at more than 70 seconds, this becomes feasible.

6.3. Core Load and Working Memory

Apart from the power and energy requirements, the requirements on core load (**LZ-OBC-OPER-003**) and working memory (RAM) (**LZ-OBC-PERF-001**) need to be considered.

6.3.1. Core load

Requirement **LZ-OBC-OPER-003** indicates that 1038 DMIPS (25%) of the core load of the Q7S needs to be available for control of locomotion operations at all times. This means that the core shall not be completely taken by hazard detection operations during operation on the Moon. However, as Figure 6.13 indicates, the four ARM cortex A7 cores of the RP2 are completely occupied by the inference with MultiResUNet, when no accelerator is attached to take away these heavy operations. Although the two ARM cortex A9 cores of the Xiphos Q7S chip are slightly more advanced and efficient, much more than the full available core load will be engaged by the hazard detection algorithm.

The control of the LMS, MCP, EDAC and PPU is of high priority. To be able to still simultaneously perform these functions, part of the core load could be left available. However, this would go at the cost of an even longer hazard detection inference time. Even at a cycle time of $150\,\mathrm{s}$ the operational cycle is completely filled by the base tasks as described in subsection 3.1.5. Thus, the cycle time would need to be prolonged to unacceptable cycle times for the CPU.

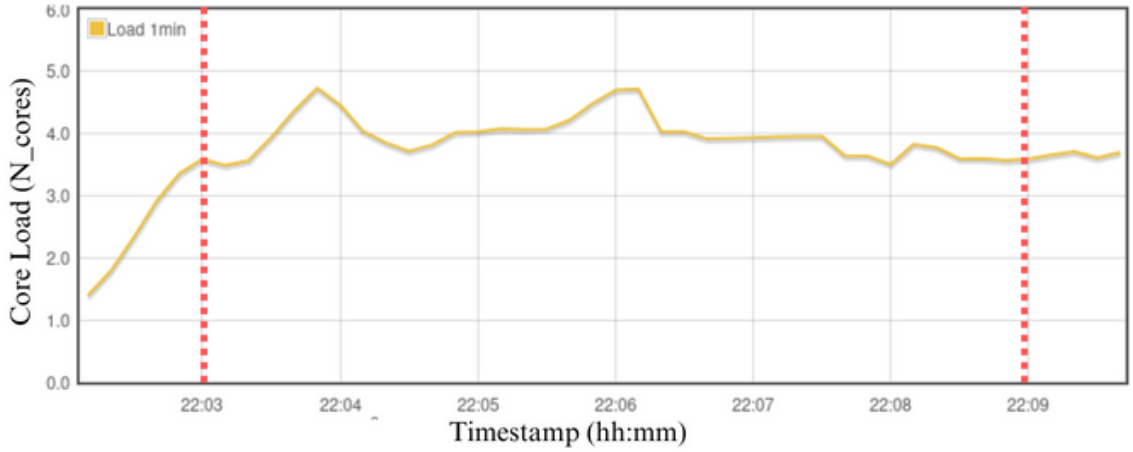


Figure 6.13: The required core load during hazard detection inference on the CPU of the RP2. The red-dotted lines indicate the start and beginning of inference. Graph obtained with RPi-Monitor [207].

In Figure 6.13, the 'Load 1 min' metric represents the average number of processes in the run queue over the last sixty seconds, offering a real-time snapshot of system demand. This figure, crucial in evaluating computational efficiency, is not confined to the number of available CPU cores in the Raspberry Pi system. It is conceivable for this load average to exceed the total core count, indicating a state where the number of active or waiting processes surpasses the immediate processing capacity of the system. Such a scenario occurs when multiple processes are concurrently vying for CPU resources, leading to a queue.

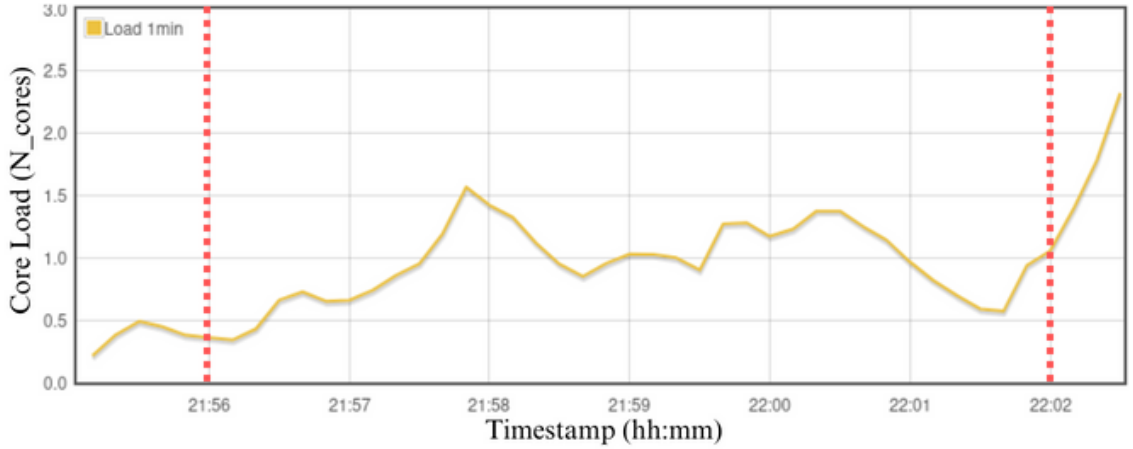


Figure 6.14: The required core load of the RP2 during preparation for inference and inference on the XE2. The red-dotted lines indicate the start and beginning of inference. Graph obtained with RPi-Monitor [207].

6.3.2. RAM

Following the specifications of the Xilinx Xiphos Q7S, the maximum available RAM is $768\,\mathrm{MB}$. The most constraining operation is inference on images, as the images will take up a large part of the available working memory. For the analysis in Figure 6.15, 10 images were preprocessed, assuming an image size of 640x480 pixels once again. For this analysis, as a worst-case scenario, it is assumed that the batch of images is preprocessed on the main OBC, so that it can be sent to the XE2 as a complete batch.

It can be seen that the maximum of the OBC's RAM capacity is almost reached at 10 images of this size. However, in this situation the 25% buffer (as demanded by requirement **LZ-OBCA-PER-001**) and a 25% communication overhead are not yet taken into account. It should be noted however, that the packages required on the RP2 OS take a large part of the available memory, and this could be optimised for implementation on Lunar Zebro. However, what can be concluded is that the working memory cannot

easily serve simultaneous (pre)processing of images, therefore complexifying the implementation of a centralized swarm.

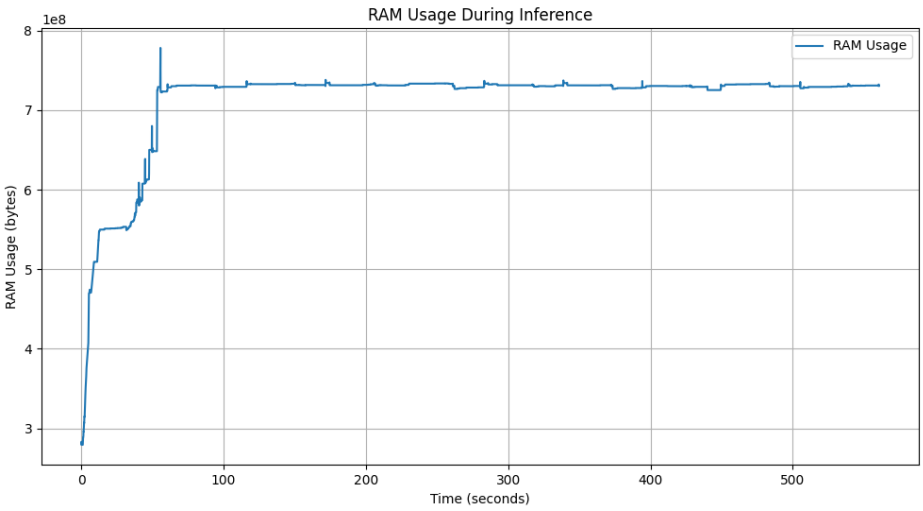


Figure 6.15: The utilized working memory for inference with 10 images of size 640x480 pixels. The images utilize 0.735 GB of the working memory.

It should be noted that with smart reworking of the image inference cycle, the required RAM can be decreased to about 370 MB. In this situation, images would be preprocessed and then inference would be ran one-by-one.

In conclusion, considering the image inference pipeline (Figure 6.1) and the possibility to run preprocess images one-by-one and run inference concurrently, the RAM requirements should be met in all scenarios. However, in the case of a centralized swarm, this would require a complexly streamlined pipeline, in which images are loaded one-by-one.

6.4. Non-volatile Memory

The non-volatile memory on the Q7S can be divided into Flash memory and the external SD drivers. The Flash memory is ideally able to save all firmware and other algorithms redundantly, as algorithms can be obtained much faster from Flash than from the SD.

6.4.1. Flash Memory

There is 256 MB of Flash available. It was discussed the relevant algorithms require approximately the following storage:

- **Firmware:** 10 MB
- **Path Planning:** 3 MB
- **FDIR:** 14.5 MB
- **Hazard Detection:** 29 MB

These are very conservative estimates. For the hazard detection algorithm, for example, the FP32 version of the tflite model is taken into account. However, the total required storage for all algorithms is *56 MB*. It can thus be concluded that the Flash memory allows for saving all relevant software in threefold (**LZ-OBCA-RAMS-005**).

6.4.2. SD Memory

The utilization of data on the non-volatile SD memory is essential for storing four primary categories of acquired data throughout the mission. For all categories, conservative estimates of the maximum required storage were conducted:

1. **A Map of the Area:** in subsection 4.3.3 it was described that an APF algorithm will be used for path planning. In Table 6.9, it is indicated that each obstacle occupies 6 bytes of storage. The total storage size required for the list of obstacles is contingent upon the number of obstacles (N_o) present. According to the model discussed in subsection 2.3.2, an area of 5x5 square meters can accommodate up to 35 obstacles (1.4 obstacle per m^2) in an area with a rock density of $5\,\%$. Knowing that the second generation of Lunar Zebro potentially has more interest in rocky regions due to the paradigm shift from radiation measurements to regolith or rock composition, the hazard abundance is multiplied by a factor of two.

   Table 6.9: Storage of obstacles

Data	Type	Size
X_o	FP16	16 bits
Y_o	FP16	16 bits
R_o	FP16	16 bits

   In the most optimistic case, the rover can move at $5\,\mathrm{cm\,s^{-1}}$. Taking into account the operational cycle and the required 3-hour recharge phase of the rover, the rover can cover $15\,120\,\mathrm{m}$ in 14 days. The maximum required storage for one individual for 14 days of operation on the Moon can be calculated to be:

   $$S_{map} = d \times w_{obs} * N_o * S_o = 15120 \times 5 \times 2.8 \times 6 = 6.4\,\mathrm{MB} \tag{6.1}$$

   where S_{map} is the maximum storage size of the map for one rover, d is the max distance covered, w_{obs} is the total assumed width that the rover can observe ($5\,\mathrm{m}$), N_o is the calculated number of obstacles per m^2 and S_o is the required storage per obstacle.

2. **The Planned Path**: following earlier research [39] the path, that will be determined with the APF algorithm, can be stored in FP16 values. The path is represented and stored as a series of coordinate pairs (X_r, Y_r). Each pair of coordinates is composed of 4 bytes, contributing to the storage size of the path data. The Bacteria-Aritificial Obstacle (B-AO) algorithm designed by Manteaux [39], would then need about $0.5\,\mathrm{kB}$ of storage for a goal location distanced at $3\,\mathrm{m}$ from the rover.

3. **Images:** images can be stored on board the rover for two reasons. Either for retrieval of an image so for redetermining the location of obstacles, or to pass on footage of locations of scientific interest to a lunar base or lander. For these purposes, the rover will not need more than 50 stored images. The images taken by the SHRIMP cameras are RGB images with 640 x 480 = 307,200 pixels. Each pixel in the image, being in RGB format, necessitates 3 bytes of storage, representing the red, green, and blue color channels. Therefore, the total storage required for this image can be calculated as 307,200 pixels multiplied by 3 bytes per pixel, totaling 921,600 bytes or roughly $1\,\mathrm{MB}$. A total storage of $50\,\mathrm{MB}$ would thus be required.

4. **Scientific Payload Data:** In section 2.1 it was discussed that besides the radiation payload designed for the first-generation rover, a laser mass spectrometer could be an interesting payload for the next generation of Lunar Zebro. Since this payload requires much more storage, this payload will be taken into account regarding considerations for non-volatile memory. The considered laser mass spectrometer [35], demands a maximum of $128\,\mathrm{kB}$ at a $2.5\,\mathrm{ns}$ time resolution. In the context of operational use, six measurements per hour in operation are assumed. Given these parameters, the total storage requirement for 14 days of rover operation is calculated at approximately $64.5\,\mathrm{MB}$, considering the cumulative data from the acquired measurements during the specified period and taking into account the solar charging periods.

5. **Housekeeping Data:** The non-volatile memory required for housekeeping data of all subsystems when saved for three days was calculated in Equation 4.1.

Table 6.10: The required non-volatile storage for the map, planned path, images and payload data

Data	Required Non-volatile Storage [MB]
Map	6.4
Path	0.50×10^{-3}
Images	50
Payload data	64
Houskeeping	0.48
Subtotal	120.9
Redundancy (3x)	362.7
Overhead (25%)	90.68
Grand Total	453.38

Table 6.10 depicts the required storage to save all data once. Following the explanation in section 4.4 and requirement **LZ-OBCA-RAMS-005** all data needs to be saved three times on the non-volatile storage. When also accounting for a 25% storage overhead, approximately **500 MB** of non-volatile storage is required. This is represented in requirement **LZ-OBCA-PERF-002**.

The figures represent the baseline storage needs, while the final design might allow for extra non-volatile storage space. This extra storage can be used for storing more high-resolution images, detailed environmental data, or additional scientific measurements, enhancing the mission's scientific output or operational efficiency. The allocation of this additional memory storage will be determined by the final design team, based on mission requirements and the rover's operational parameters. This flexibility ensures a design that is adaptable and capable of maximizing the scientific and operational effectiveness of the lunar rover mission.

6.5. Communication Bandwidth

In subsection 3.1.7 the available communications bandwidth for the nano rover was discussed, whilst the communication protocol and the accessory package overhead were discussed in subsection 4.3.5. In this section, the possibility of a centralized swarm will be discussed, by means of analysis of the data that needs to be transmitted, and the available bandwidth.

In Figure 4.11 centralized and decentralized swarms are depicted. A major difference is the communication requirement. Following the assumptions in subsection 3.1.4, a decentralized swarm would pose much tighter requirements on the communication subsystem than the centralized swarm would. First of all, every rover would have to send all its data for inference to the computational hub. In a decentralized swarm, individuals only share their location and if applicable, locations of interest. In a centralized swarm, all images and housekeeping data would need to be sent over to the computational hub.

Table 6.11 summarizes the required data that are required to be sent to other rovers every operational cycle. The locations of the rover itself and of an object of scientific interest each require about 8 bytes, assuming 32-bit floating point precision and a latitude and longitude coordinate. Furthermore, if the computational hub would need to perform AI-based FDIR on the housekeeping data of the other rovers in the swarm, all housekeeping data would need to be transferred to the computational hub. The housekeeping data for a period of one hour is estimated to be 2880 Bytes, as explained in section 6.4. Finally, depending on whether monocular vision is implemented, one or two images would need to be sent to the computational hub for inference. The data size of the images dominates the size of the data packages that need to be sent to the computational hub.
The computational hub would need to receive and process a multiple of the total sent data of an individual, depending on the number of rovers dependent on one computational hub.
Where the communication bandwidth (250 kbit/s) as determined in subsection 4.3.5 easily allows for the communication in a decentralized swarm, one image of 1 MB would require approximately 40 seconds to be transferred to the computational hub, assuming the use of Zigbee communication system and

Table 6.11: Outbound transferred data in two different swarm designs.

Data Type	Decentralized [Bytes]	Centralized [Bytes]
Location	8	8
Scientific Interest	8	8
Housekeeping	N/A	2,880
Image	N/A	1 or 2 $\times$ 10^6
Total	16	$\approx$ 1 or 2 $\times$ 10^6

protocol, and assuming the images are transferred at original size (640x480 pixels, RGB).
The max data rate of rs485 is 10 Mbit/s, and this allows for sending 1 image per second, thus in a time span of 10 seconds, the image data of 5 rovers could be transmitted throughout the data bus. Therefore, the RS485 data bus can actually support the centralized swarm concept, although the communication system would need a thorough redesign, and would require much more power and energy.

6.6. System Verification

The system requirements for the OBCA were stated in subsection 3.1.3. Following the research performed, this section aims to verify these requirements. Actual validation of the system was considered out of scope for this research, as this would entail implementation of the OBCA on a lunar testbed. However, this will be recommended for further research in chapter 8.

Table 6.12: Lunar Zebro OBC system requirements

System Req,	Verified	Justification
FUN-001	Yes	Scientific data is accounted for in terms of non-volatile memory, the operational schedule, and even DL-based inference on this data for autonomous mission planning is discussed.
FUN-002	Yes	DL-based hazard detection is researched and implemented, and a network with relatively high accuracy and a low number of false negatives is selected (please refer to subsection 5.1.3).
FUN-003	Yes	APF path planning accounted for, without required intervention by operators. All analyses are performed for this operational cycle time, and the influence of changing cycle time has also been analysed.
FUN-004	Yes	DL-based FDIR on housekeeping is accounted for, using TCNs.
FUN-005	Yes	The connection with the PPU allows for power cycling when one core detects a fault in the other. The PPU can also take over minimal functionalities of the central OBC.
FUN-006	Yes	The connection with the PPU allows for power cycling of all subsystems.
DES-001	Yes	The peak power amounts to 5.3 Watt.
DES-002	Yes	Energy required within energy budget of operational cycle. Sensitivity analysis performed for understanding of budgetary changes.
DES-003	Yes	Both boards have gone through TVAC tests [30, 146].
DES-004	Yes	Both boards have gone through TVAC tests [30, 146].
DES-005	Yes	By inspection.
DES-006	Yes	The multicore PS of the Zynq 7020 allows for this.

Table 6.12: Lunar Zebro OBC system requirements

System Req,	Verified	Justification
PERF-001	Yes	For decentralized swarm easily. For a centralized system, the OBCA for the computational hub can support inference for up to 2 *other* individuals.
PERF-002	Yes	By analysis in section 6.4.
INT-001	Yes	Inspection, meets requirement by design.
INT-002	Yes	Inspection, meets requirement by design.
INT-003	Yes	Inspection, meets requirement by design.
INT-004	Yes	Inspection, meets requirement by design.
INT-005	Yes	By Inspection.
INT-006	Yes	Inspection, meets requirement by design.
RAMS-001	Yes	TID well below max thresholds for devices.
RAMS-002	Yes	By inspection.
RAMS-003	T.B.C.	XE2 has been tested up to 65°, would need further consideration.
RAMS-004	T.B.C.	The Myriad chip has been tested against the correct radiation conditions. The use of the Q7S in other lunar missions and work described in subsection 4.4.3 suggests that the Q7S is protected against latch-ups as well, but further research is required for final verification.
RAMS-005	Yes	In section 6.4 it was described that the non-volatile data storage of the Q7S would be sufficient, even for triple redundancy.
RAMS-006	Yes	section 4.1 accounts for the complete coverage of the extra avionics.
RAMS-007	Yes	By design, also taken into account in radiation analysis in section 4.4.
OPER-001	Yes	Operational assumption.
OPER-002	Yes	Operational assumption.
OPER-003	Yes	This computational power becomes available due to the XE2.
OPER-004	Yes	The system is designed to iterate its operational cycle every 0.8 meters. Elaborate sensitivity analysis performed in subsection 6.2.3.

7

Conclusion

The main research question for this project is: *How could the OBCA for the next-generation Lunar Zebro be designed cost-effectively?* The 'next-generation' Lunar Zebro was envisioned to have advanced capabilities, including DL-based hazard detection and FDIR on housekeeping data, autonomous mission planning, and functioning in an exploratory swarm. To answer this main question, a systems engineering approach was followed, adhering to the V-model methodology.

Stakeholders were identified and their requirements were stated. Stakeholder requirements **LZ-OBCA-SH-008** and **LZ-OBCA-SH-010** dictated the implementation of DL-based hazard detection and FDIR on subsystem housekeeping data. This process ultimately resulted in the definition of technical system requirements, which were presented in Table 6.12. Throughout the work, these requirements dictated the design choices made. The *Design* requirements related to power and energy **LZ-OBCA-DES-001** and **LZ-OBCA-DES-002** received extra attention in chapters 5 and 6, due to the inherently stringent SWaP constraints associated with nano-rovers such as Lunar Zebro. Other than the *Design* requirements related to energy and power, the *Operational* and *Performance* requirements were verified by analysis in subsection 6.2.2 and section 6.3. The *rams* requirements related to radiation also required analysis for verification. The *Interface* requirements were taken into account in the detailed design phase, during the conception of the connections with the EPS and the design of the data bus.

Following the requirements, the operational details and assumptions were discussed. These formed the foundation of this research, as the operations of the rover on the Moon determined how constraining the energy and data budgets were for the OBCA. Therefore, a sensitivity analysis was performed in subsection 6.2.2.

As a final step in the conceptual design, 5 design options were conceived. An initial comparison of these design options was performed, and two final design options were determined feasible for the execution of the tasks described in subsection 3.1.5, while adhering to the system requirements:

1. A SBC
2. A SBC, with an Additional VPU Board

Because Lunar Zebro indicated a preference for the use of the Xiphos Q7S as the main OBC, the Xiphos Q7S was considered as the main OBC in the architecture. Ubotica's CogniSat XE2 board, housing a Myriad X VPU, was considered as the AI-accelerating board.

Having conceived the design options, these were worked out in more detail. The extra mass and cost of bringing the accelerator to the Moon were calculated; per rover, an extra $300\,\mathrm{g}$ and *$390,000* were required. The detailed design of the OBCA also involved the design of the data bus. The RS485 data bus consisted of 5 parallel buses. Moreover, the central OBC and the VPU board were connected through Ethernet (data and firmware transmission), where a CAN connection (XE2 telemetry, command & control) is highly advised. Furthermore, the design of the OBCA involved a precise understanding of the

required software. Therefore, the most demanding algorithms were designed, and specifications when run on the PS of the Zynq 7020 for these algorithms were determined. These specifications, together with the different operational modes, were used later during the endurance test simulations in chapter 5. As the swarming method also determined the requirements on the OBCA, this was also worked out to a level of detail sufficient to understand the requirements that it would pose on the OBCA. Finally, the detailed design incorporated an assessment of the radiation risk. The SEU rate and TID were taken into account for both considered devices. Despite employing a highly conservative approach, the projected upset rates for both devices were less than one incident, which would not compromise the mission objectives. A SEU handling strategy was presented; the design options with the XE2 attached can expect even less critical upsets.

To understand the implications of running DL-based hazard detection on Lunar Zebro, a relatively lightweight UNet-variant was trained and implemented for testing. Both design options and different operational modes were considered. Every step in the inference pipeline was carefully timed, and power measurements were performed. The results of these tests are summarized in Table 6.1. These measurements were used in an endurance test simulation. This simulation emulated the operational cycle of the OBCA, and considered its power and energy requirements. It was found that the design option without the AI-accelerator could not operate within the initially assumed design space. Because of the operational assumptions discussed, the energy budget, cycle time, and inference times were varied to understand the consequences of changes in the design space. Finally, the different design options for the swarm were considered. The detailed findings and considerations resulting from these simulations are discussed in section 7.1.

System verification was performed for the design option with the CogniSat XE2 board. It was found that this design option meets almost all requirements, with both monocular and stereo vision, albeit at the cost of bringing an extra board to the Moon. Only requirements **LZ-OBCA-RAMS-003** and **LZ-OBCA-RAMS-004** require further investigation.

7.1. Research Questions

To conclude, the main research question was supported by three subquestions as introduced in chapter 1, for which the concluding analyses were as follows:

What are the required resources for DL-based hazard detection, and how could the OBCA design under investigation provide these resources (respecting the SWaP constraints)?

The specifications of DL-based image segmentation with MultiResUNet on the SHRIMP data were summarized in Table 6.2. A more detailed view of the image and inference pipeline, combined with the required resources for both design options, is found in Table 6.1. Implications on the core load and different types of memory are discussed in section 6.3.

It was found that, given the constrained resources of the nano-rover and the constraining cycle time, an AI-accelerator was required to speed up the demanding computations for hazard detection. The ARM Cortex A7 quad-core PS of the RP2, deemed comparable to the PS of the Xiphos Q7S, was not able to run hazard detection within the available time. Only at much higher cycle times, would a design option without an accelerator have been able to operate, as portrayed in Table 6.5.

How can the OBCA facilitate DL-based FDIR, swarming, path planning, and autonomous mission planning?

The endurance test described in section 6.2 was executed to understand the performance of the different design options when performing all named tasks. This simulation answered whether the design options adhered to the requirements related to i) peak power draw (**LZ-OBCA-DES-001**), ii) energy budget (**LZ-OBCA-DES-002**), iii) time constraints (**LZ-OBCA-OPER-004**).

When adhering to the main assumptions, Table 6.3 displays the relevant results. The design option without the XE2 did not suffice, because of time and energy constraints. At 10% smaller energy budgets, no design option would suffice, so the energy budget could not be decreased, as summarized in Table 6.4. At cycle times above 90 seconds, the single-board design option would become an option. However, higher cycle times imply larger covered distances before detecting hazards on new images and replanning paths. Increasing the cycle times extensively could lead to a loss of efficiency of the planned paths, and increase the chance of collision. At 90 seconds, the cycle time was more than double the original cycle time (40 seconds). Moreover, requirement **LZ-OBCA-OPER-003** was still violated. This requirement could eventually have been met by spreading out the inference time even further while saving some computational power. Therefore, the single-board design option was considered infeasible.

Moreover, at cycle times up to 40 seconds, the use of the warm-boot mode was advised. Above this cycle time, the higher 'base' power overruled the positive consequences of the warm-boot mode, deeming this mode unsuccessful.

When choosing the design option with the AI-accelerator, there are multiple ways to leverage its power. The use of DL-based hazard detection and DL-based FDIR on housekeeping data of subsystems were implied during the analysis above. However, for fully autonomous mission planning, the rover could have used a different CNN to classify rocks of interest for further scientific analysis. This would have required the cycle time to increase to 80 seconds for stereo vision, and only to 50 seconds with monocular depth perception implemented. This showed that if monocular depth estimation could be implemented, there are many opportunities to leverage the newly available energy.

Finally, the OBCA can easily meet the memory requirements of the rover and its advanced functionalities. This is primarily due to the relative affordability of memory in COTS products. In this project, it was shown that a COTS-based system can comfortably accommodate the memory needs of an advanced rover. However, memory considerations become more complex in a centralized swarm design, necessitating careful planning and execution.

Should the OBCA be designed homogeneously across the whole swarm?

Figure 6.11 depicts the energy consumption of the OBCA with XE2 for the computational hub in a centralized swarm. Moreover, Table 6.6 and Table 6.7 summarize the energy consumption of the OBCA for the computational hub in a centralized swarm when running inference for a different amount of rovers. These depictions show that at higher cycle times (at least 60 seconds), the design option with the XE2 did allow for this when considering energy consumption. However, the RAM use needed to be carefully considered. Figure 6.15 shows the utilized RAM when 10 images were loaded to memory. At 735 MB, this would fill up the complete RAM storage of the Q7S. Considering the 25% communication overhead, the buffer, and housekeeping & location data that need to be sent to and processed by the computational hub, a maximum of 3 rovers could rely on one computational hub, and only with very careful implementation of the communication protocol between the different rovers, as the cycle time would be a constraining factor.

In summary, two primary concerns arose with a centralized swarm design.

1. The communication subsystem would need a thorough redesign, as explained in section 6.5. The computational hub would require a much higher data rate than the Zigbee communication subsystem that was currently implemented on the Ingenuity helicopter, which was taken as an example of close-range, low-power communication. This redesigned communication subsystem would have gone at the cost of extra energy use, which in turn would have triggered a 'snowball effect' of mass accumulation.

2. A centralized swarm configuration would compromise the swarm's robustness. Multiple units would depend on a single computational hub, creating a vulnerability. This centralized approach would reduce adaptability to unforeseen circumstances and overall robustness. While in a swarm

'the strength is in the numbers', this no longer holds when a centralized swarm design is chosen. Even in the likely situation that communication signals are blocked, multiple individuals would lose their ability to detect hazards and thus plan a path autonomously.

Designing the rover for a centralized swarm, and thus with heterogeneous OBCA architecture, is thus a deliberate choice against the actual reasons that a swarm was adopted in the first place. A decentralized swarm design was thus considered the most feasible design option, albeit at a higher mission cost.

8

Recommendations for Further Work

Based on the findings discussed in the preceding chapter, this chapter presents a set of recommendations for future research and development for the OBCA and algorithms of Lunar Zebro and lunar nano-rovers in general. The following recommendations are given:

- Should Lunar Zebro eliminate the necessity for the implementation of algorithms on the FPGA, the focus could shift towards a central SBC with a sole CPU. This modification potentially reduces energy consumption, allowing for an extended operational phase and enhanced productivity of the rover. A pertinent example is the ISIS Space iOBC [208], which demonstrates a nominal power consumption of 400 mW, a significant reduction from the 2 W required by the current Q7S configuration. This energy efficiency would directly contribute to increased rover endurance and exploration capabilities.

- Optimize and rigorously test the implementation of the hazard detection algorithm on the On-Board Computer (OBC). Focus on refining both the image preprocessing pipeline and the underlying model, with particular emphasis on leveraging the XE2 accelerator for enhanced efficiency. Additionally, explore the integration of mixed precision methods, as proposed by Micikevicius et al. [195], to potentially boost the algorithm's accuracy. This optimization effort holds the potential to significantly improve both performance and precision.

- Exploring a broader range of segmentation algorithms, particularly those optimized for on-edge applications, is crucial. These algorithms, currently employed in edge devices on Earth like mobile phones, offer potential benefits for space applications. This study employed a segmentation network tested for rock detection in extraterrestrial settings. However, the effectiveness of several alternative architectures, such as MobileNet [209], MobileNetv2 [210], and SeResNet18 [211], in rock segmentation remains to be validated. These architectures are designed for low-SWaP applications, making them appealing for space missions.

  Investigating these algorithms' capabilities in rock segmentation is a critical next step. Such research should assess how these algorithms perform across various OBCAs, as was done in this project. For applications utilizing these more lightweight algorithms, the primary concern will be maintaining high accuracy, especially for a rover that cannot surpass obstacles larger than 3 cm. However, if very high accuracy can be reached with the lightweight networks, then a SBC like the ISIS Space iOBC might be sufficient to run DL-based inference. Finally, it would also get easier to implement the algorithm on a constrained FPGA device.

- Consider implementing the DL-based algorithms on a FPGA like the Zynq 7020, taking into account the tradeoff between implementation effort, cost and performance. Especially in combination with the exploration of more lightweight algorithms. This will cost substantial engineering time, and even then the resources this would require of boards like the Q7s is a concern, but it is

possible [3]. The implementation would require deep expertise, as it is a very complex endeavour. If this would work out, the Xiphos Q7S board would suffice for operation, and at very high efficiency [FPS/W] and frame rate [FPS] [142].

- Creating an extensive lunar testbed to validate the operation of the rover in a simulated environment. During validation runs on an actual test bed, the hazard detection capabilities, path planning efficiency and the energy use of different subsystems can be tracked simultaneously.

  Moreover, this testbed can be used to train a CNN, like MultiUResNet, enhancing its capabilities to accurately detect craters and slopes. This task requires a significant commitment of man-hours for the labeling of the dataset. The process involves the development and verification of the test bed, the acquisition of images for labeling, and the actual labeling process itself. While this represents a considerable investment both financially and in terms of labor, the resulting detailed and diverse dataset could markedly improve the CNN's accuracy in lunar terrain interpretation and navigation.

- Implement monocular depth estimation for hazard detection using the same lunar test bed. This approach involves training the network to estimate distances to various hazards, enabling the recognition of rocks and determination of their distances in a single integrated process. This method, as proposed by Khan et al. [66], leverages DL techniques to enhance the rover's navigational capabilities by accurately assessing the spatial relationship between the rover and potential hazards.

- Implement the remaining algorithms during the advanced stages of the design process, after the completion of all software design. More details about the actual algorithms allow for a more detailed endurance test with different OBC options. However, development of these algorithms will take years, therefore this study gives a very good initial overview of what OBCA would be required to answer the demands of the stakeholders.

- Testing the implemented algorithms for radiation, to understand how the different algorithms and the operating system would exactly respond to radiation and SEUs when in operation. The exact MCP implementation is of utmost important here, as the places where algorithms are stored, and the SEU cross-section of those devices determine how often SEUs occur in different algorithms and how critical these SEUs are. Together with these tests, EDAC methods can be improved and customised.

  An interesting hypothesis could be that the DL-based networks can be trained to perform well, even when bits in the imaging data are flipped, or when pixels of the camera are broken. Furthermore, the algorithm should be tested and prepared for handling situations like the recognition of other rovers appearing from behind obstacles or even scenarios where it captures its own shadow in the images. These additional tests ensure that the algorithm exhibits adaptability and robustness, making it well-equipped to handle a wide range of challenging conditions that may arise during lunar missions.

- Explore and implement the utilization of Convolutional Neural Networks (CNN) for the identification of scientifically significant objectives. This involves training CNNs on lunar images encompassing various rock types and augmented datasets to enable classification of diverse rocks. Additionally, it could be considered to leverage features such as rock shapes, colors, sizes, and the context of surroundings (maria/highlands) to enhance the accuracy of the analysis.

- Re-run the power tests, previously conducted on the Raspberry Pi 2, on the Xilinx Xiphos Q7S PS. This effort promises to yield significantly more accurate power and energy measurements. Given the ongoing Linux implementation work on this board, these tests can be performed this year already.

- Verify the feasibility of the current assumption that intra-swarm communication takes place during the final phase of the operational cycle. To accomplish this, a decision regarding the communication system's design is imperative. Subsequently, establish a validation setup that simulates multiple communicating agents. Determine whether all messages can be transmitted within the allocated time frame and assess whether the computational hub has sufficient time for inference tasks.

- Conduct thorough validation and testing of the data bus operation. This step involves comprehensive assessment and verification of the data bus's functionality to ensure reliable and efficient data transfer within the system.

- The low-power mode discussed in this thesis needs to be carefully implemented and tested. For this, it is important to test how well the rover can maintain the operational cycle as discussed in subsection 3.1.5. The better this is possible, the higher the chance that the Ethernet connector on the XE2 can be left in a passive OFF state during most of the cycle.

References

[1] Institute of Technical Mechanics of the National Academy of Science of Ukraine and the State Space Agency of Ukraine, Dnipropetrovsk, Ukraine et al. “Deep learning for spacecraft guidance, navigation, and control”. en. In: *Kosmična nauka ì tehnologìâ* 27.6 (2021), pp. 38–52. ISSN: 15618889, 25181459. DOI: 10.15407/knit2021.06.038. URL: http://space-scitechjournal.org.ua/en/archive/2021/6/03 (visited on 01/10/2024).

[2] Haiqiang Liu et al. “RockFormer: A U-Shaped Transformer Network for Martian Rock Segmentation”. In: *IEEE Transactions on Geoscience and Remote Sensing* 61 (2023). Conference Name: IEEE Transactions on Geoscience and Remote Sensing, pp. 1–16. ISSN: 1558-0644. DOI: 10.1109/TGRS.2023.3235525.

[3] Matt Cross et al. “Enabling Autonomy and Operations for Lunar Surface Missions: An Overview of Demonstrated Capabilities”. en. In: (2023).

[4] Nabil Ibtehaz and M. Sohel Rahman. “MultiResUNet : Rethinking the U-Net architecture for multimodal biomedical image segmentation”. en. In: *Neural Networks* 121 (Jan. 2020), pp. 74–87. ISSN: 08936080. DOI: 10.1016/j.neunet.2019.08.025. URL: https://linkinghub.elsevier.com/retrieve/pii/S0893608019302503 (visited on 07/14/2023).

[5] Qi Wang and Yan Piao. “Depth estimation of supervised monocular images based on semantic segmentation”. en. In: *Journal of Visual Communication and Image Representation* 90 (Feb. 2023), p. 103753. ISSN: 10473203. DOI: 10.1016/j.jvcir.2023.103753. URL: https://linkinghub.elsevier.com/retrieve/pii/S1047320323000032 (visited on 01/01/2024).

[6] James Murphy, John E Ward, and Brian Mac Namee. “Machine Learning in Space: A Review of Machine Learning Algorithms and Hardware for Space Applications□”. en. In: (2021). URL: https://ceur-ws.org/Vol-3105/paper21.pdf.

[7] Antonio Carlos Cob-Parro et al. “Hardware alternatives for the implementation of machine learning systems applied to image processing”. en. In: (Sept. 2021). URL: https://www.researchgate.net/profile/Alfredo-Gardel/publication/369730641_Hardware_alternatives_for_the_implementation_of_machine_learning_systems_applied_to_image_processing/links/642956d692cfd54f844a0766/Hardware-alternatives-for-the-implementation-of-machine-learning-systems-applied-to-image-processing.pdf.

[8] Danish Ali, Anees Ur Rehman, and Farhan Hassan Khan. “Hardware Accelerators And Accelerators For Machine Learning”. In: *2022 International Conference on IT and Industrial Technologies (ICIT)*. Oct. 2022, pp. 01–07. DOI: 10.1109/ICIT56493.2022.9989124.

[9] Gianluca Giuffrida et al. “The Φ-Sat-1 Mission: The First On-Board Deep Neural Network Demonstrator for Satellite Earth Observation”. en. In: *IEEE Transactions on Geoscience and Remote Sensing* 60 (2022), pp. 1–14. ISSN: 0196-2892, 1558-0644. DOI: 10.1109/TGRS.2021.3125567. URL: https://ieeexplore.ieee.org/document/9600851/ (visited on 01/10/2024).

[10] *Homepage | Ubotica Technologies*. en-US. URL: https://ubotica.com/ (visited on 05/31/2023).

[11] *CogniSAT-XE2 Product Overview*. en. URL: https://ubotica.com/product/cognisat-xe2-product-overview/ (visited on 12/16/2023).

[12] *Intel® Movidius™ Myriad™ 2 Vision Processing Unit 4GB - Product Specifications*. en. URL: https://www.intel.com/content/www/us/en/products/sku/122461/intel-movidius-myriad-2-vision-processing-unit-4gb/specifications.html (visited on 12/16/2023).

[13] Steve Creech, John Guidi, and Darcy Elburn. “Artemis: An Overview of NASA’s Activities to Return Humans to the Moon”. en. In: *2022 IEEE Aerospace Conference (AERO)*. Big Sky, MT, USA: IEEE, Mar. 2022, pp. 1–7. ISBN: 978-1-66543-760-8. DOI: 10.1109/AERO53065.2022.9843277. URL: https://ieeexplore.ieee.org/document/9843277/ (visited on 03/30/2023).

[14] *Peregrine-Payload-Users-Guide.pdf*. URL: https://www.astrobotic.com/wp-content/uploads/2021/01/Peregrine-Payload-Users-Guide.pdf (visited on 11/14/2023).

[15] Brett Tingley published. *Mexico's 1st moon mission will send 5 tiny robots aloft on Peregrine lunar lander Jan. 8*. en. Jan. 2024. URL: https://www.space.com/peregrine-lunar-lander-mexico-colmena-micro-robots (visited on 01/11/2024).

[16] *Lunar Zebro | Nano Rover TU Delft*. en-US. URL: https://zebro.space/ (visited on 12/13/2023).

[17] J. M. Muñoz Tejeda et al. "Environmental analysis of nanorovers in a swarm for lunar scientific missions: 70th International Astronautical Congress, IAC 2019". In: 2019. URL: http://www.scopus.com/inward/record.url?scp=85079163065&partnerID=8YFLogxK (visited on 12/13/2023).

[18] FrischaufNorbert et al. "NewSpace: New Business Models at the Interface of Space and Digital Economy: Chances in an Interconnected World". en. In: *New Space* (June 2018). Publisher: Mary Ann Liebert, Inc. 140 Huguenot Street, 3rd Floor New Rochelle, NY 10801 USA. DOI: 10.1089/space.2017.0028. URL: https://www.liebertpub.com/doi/10.1089/space.2017.0028 (visited on 01/11/2024).

[19] *Lunar Zebro*. nl-NL. URL: https://www.tudelft.nl/innovatie-impact/innovation-projects/lunar-zebro (visited on 01/13/2024).

[20] *Moon Mission | Lunar Zebro*. URL: https://zebro.space/missions/moon_mission/ (visited on 05/04/2023).

[21] J B Miog. "Design of the locomotion subsystem for Lunar Zebro". en. PhD thesis. Delft: Delft University of Technology, Aug. 2018. URL: https://repository.tudelft.nl/islandora/object/uuid%3A0b69bb78-38c3-442e-b416-6bae4545e6e0.

[22] Liang Sim, M.L. Cummings, and Cristin A. Smith. "Past, present and future implications of human supervisory control in space missions". en. In: *Acta Astronautica* 62.10-11 (May 2008), pp. 648–655. ISSN: 00945765. DOI: 10.1016/j.actaastro.2008.01.029. URL: https://linkinghub.elsevier.com/retrieve/pii/S0094576508001045 (visited on 12/13/2023).

[23] Jamison Heard, Prakash Baskaran, and Julie A. Adams. "Predicting task performance for intelligent human-machine interactions". en. In: *Frontiers in Neurorobotics* 16 (Sept. 2022), p. 973967. ISSN: 1662-5218. DOI: 10.3389/fnbot.2022.973967. URL: https://www.frontiersin.org/articles/10.3389/fnbot.2022.973967/full (visited on 12/13/2023).

[24] Robert W. Thompson et al. "Mass growth in space vehicle and exploration architecture development". en. In: *Acta Astronautica* 66.7-8 (Apr. 2010), pp. 1220–1236. ISSN: 00945765. DOI: 10.1016/j.actaastro.2009.10.013. URL: https://linkinghub.elsevier.com/retrieve/pii/S0094576509004998 (visited on 12/22/2023).

[25] Dan Friedlander. *COTS in Space*. en-US. Dec. 2019. URL: https://www.doeeet.com/content/eee-components/cots-new-space-leo-constellations/cots-in-space/ (visited on 12/13/2023).

[26] Jason James. *Detailed Design Document Lunar Zebro*. 2021.

[27] *NCR18650B-Panasonic.pdf*. URL: https://z3d9b7u8.stackpathcdn.com/pdf-down/N/C/R/NCR18650B-Panasonic.pdf (visited on 05/11/2023).

[28] Jennifer A Herman and Marshall C Smart. *Operation of Lithium-Ion Batteries in the Extreme Environments of Mars*. en. University of Maryland, Dec. 2021. URL: https://creb.umd.edu/sites/creb.umd.edu/files/HermanMarshall-10DEC2021-CREB.pdf.

[29] B.V. Ratnakumar et al. "Performance characteristics of lithium-ion cells for Mars sample return Athena Rover". In: *Collection of Technical Papers. 35th Intersociety Energy Conversion Engineering Conference and Exhibit (IECEC) (Cat. No.00CH37022)*. Vol. 1. July 2000, 638–645 vol.1. DOI: 10.1109/IECEC.2000.870848. URL: https://ieeexplore.ieee.org/document/870848 (visited on 01/15/2024).

[30] *XTI-2001-2020-f-Q7S-Spec-Sheet.pdf*. URL: https://xiphos.com/wp-content/uploads/2015/06/XTI-2001-2020-f-Q7S-Spec-Sheet.pdf (visited on 05/04/2023).

[31] *Q7S - Satellite Command & Data Handling | AAC SpaceQuest*. en-GB. URL: https://www.aac-clyde.space/what-we-do/space-products-components/command-data-handling/q7s (visited on 12/13/2023).

[32] Deon Reynders, Steve Mackay, and Edwin Wright. "RS-485 overview". In: *Practical Industrial Data Communications*. Dec. 2004. URL: https://www.researchgate.net/publication/344938324_5a_RS-485_overview (visited on 12/21/2023).

[33] Colin Walls. "Multicore Embedded Systems". en. In: *Embedded Software*. Elsevier, 2012, pp. 365–382. ISBN: 978-0-12-415822-1. DOI: 10.1016/B978-0-12-415822-1.00010-6. URL: https://linkinghub.elsevier.com/retrieve/pii/B9780124158221000106 (visited on 12/31/2023).

[34] Abhimanyu Shanbhag. "REDMOON: Radiation Environment and Dose Monitoring On-board a Nano-Rover". en. PhD thesis. Delft: TU Delft, Nov. 2022.

[35] Urs Rohner et al. "A highly miniaturised laser ablation time-of-flight mass spectrometr for planetary rover". In: *Review of Scientific Instruments - REV SCI INSTR* 75 (May 2004). DOI: 10.1063/1.1711152.

[36] Mingchuan Wei et al. "Design and flight results of the VHF/UHF communication system of Longjiang lunar microsatellites". en. In: *Nature Communications* 11.1 (July 2020), p. 3425. ISSN: 2041-1723. DOI: 10.1038/s41467-020-17272-8. URL: https://www.nature.com/articles/s41467-020-17272-8 (visited on 02/28/2023).

[37] J. Balaram, MiMi Aung, and Matthew P. Golombek. "The Ingenuity Helicopter on the Perseverance Rover". en. In: *Space Science Reviews* 217.4 (May 2021), p. 56. ISSN: 1572-9672. DOI: 10.1007/s11214-021-00815-w. URL: https://doi.org/10.1007/s11214-021-00815-w (visited on 06/21/2023).

[38] Lars Gelling. "Path planning for Lunar rovers". en. PhD thesis. Delft University of Technology, Jan. 2023.

[39] Thomas Manteaux. "Path Planning for Lunar rovers". English. PhD thesis. Delft: Ecole Polyytechnique Federale de Lausanne, Aug. 2023.

[40] *ispace_PayladUserGuide_v2_202001.pdf*. URL: https://www.mach5lowdown.com/wp-content/uploads/PUG/ispace_PayladUserGuide_v2_202001.pdf (visited on 11/14/2023).

[41] Xiaogang Wang. "Deep Learning in Object Recognition, Detection, and Segmentation". en. In: *Foundations and Trends® in Signal Processing* 8.4 (2016), pp. 217–382. ISSN: 1932-8346, 1932-8354. DOI: 10.1561/2000000071. URL: http://www.nowpublishers.com/article/Details/SIG-071 (visited on 12/24/2023).

[42] Ritesh Kanjee. *Object Segmentation vs. Object Detection - Which One Should You Use? | LinkedIn*. June 2022. URL: https://www.linkedin.com/pulse/object-segmentation-vs-detection-which-one-should-you-ritesh-kanjee/ (visited on 12/16/2023).

[43] Niall O'Mahony et al. "Deep Learning vs. Traditional Computer Vision". en. In: *Advances in Computer Vision*. Ed. by Kohei Arai and Supriya Kapoor. Vol. 943. Series Title: Advances in Intelligent Systems and Computing. Cham: Springer International Publishing, 2020, pp. 128–144. ISBN: 978-3-030-17794-2 978-3-030-17795-9. DOI: 10.1007/978-3-030-17795-9_10. URL: http://link.springer.com/10.1007/978-3-030-17795-9_10 (visited on 12/24/2023).

[44] Nati Ofir and Jean-Christophe Nebel. *Classic versus deep learning approaches to address computer vision challenges*. en. arXiv:2101.09744 [cs]. Oct. 2021. URL: http://arxiv.org/abs/2101.09744 (visited on 12/24/2023).

[45] Shadi Mahmoodi Khaniabadi et al. "Comparative Review on Traditional and Deep Learning Methods for Medical Image Segmentation". en. In: *2023 IEEE 14th Control and System Graduate Research Colloquium (ICSGRC)*. Shah Alam, Malaysia: IEEE, Aug. 2023, pp. 45–50. ISBN: 9798350346237. DOI: 10.1109/ICSGRC57744.2023.10215402. URL: https://ieeexplore.ieee.org/document/10215402/ (visited on 12/24/2023).

[46] Ibrahim Ar Rushood et al. "Segmentation of X-Ray Images of Rocks Using Deep Learning". en. In: *Day 4 Thu, October 29, 2020*. Virtual: SPE, Oct. 2020, D041S042R002. DOI: 10.2118/201282-MS. URL: https://onepetro.org/SPEATCE/proceedings/20ATCE/4-20ATCE/Virtual/449821 (visited on 12/24/2023).

[47] Z. Hou, D. Cao, and Q. Liu. “Segmentation of Digital Rock Images Guided by Edge Feature Using Deep Learning”. en. In: *82nd EAGE Annual Conference & Exhibition*. Online, European Association of Geoscientists & Engineers, 2021, pp. 1–5. DOI: 10.3997/2214-4609.202113323. URL: https://www.earthdoc.org/content/papers/10.3997/2214-4609.202113323 (visited on 12/24/2023).

[48] Ramon Gonzalez, Dimi Apostolopoulos, and Karl Iagnemma. “Slippage and immobilization detection for planetary exploration rovers via machine learning and proprioceptive sensing”. en. In: *Journal of Field Robotics* 35.2 (Mar. 2018), pp. 231–247. ISSN: 1556-4959, 1556-4967. DOI: 10.1002/rob.21736. URL: https://onlinelibrary.wiley.com/doi/10.1002/rob.21736 (visited on 12/24/2023).

[49] Rahul Moghe and Renato Zanetti. “A Deep Learning Approach to Hazard Detection for Autonomous Lunar Landing”. en. In: *The Journal of the Astronautical Sciences* 67.4 (Dec. 2020), pp. 1811–1830. ISSN: 0021-9142, 2195-0571. DOI: 10.1007/s40295-020-00239-8. URL: https://link.springer.com/10.1007/s40295-020-00239-8 (visited on 12/24/2023).

[50] Xueming Xiao et al. “Safe Mars landing strategy: Towards lidar-based high altitude hazard detection”. en. In: *Advances in Space Research* 63.8 (Apr. 2019), pp. 2535–2550. ISSN: 02731177. DOI: 10.1016/j.asr.2019.01.005. URL: https://linkinghub.elsevier.com/retrieve/pii/S0273117719300134 (visited on 05/11/2023).

[51] R. Castano et al. “Current results from a rover science data analysis system”. en. In: *2005 IEEE Aerospace Conference*. Big Sky, MT, USA: IEEE, 2005, pp. 356–365. ISBN: 978-0-7803-8870-3. DOI: 10.1109/AERO.2005.1559328. URL: http://ieeexplore.ieee.org/document/1559328/ (visited on 03/21/2023).

[52] David R. Thompson and Rebecca Castano. “Performance Comparison of Rock Detection Algorithms for Autonomous Planetary Geology”. en. In: *2007 IEEE Aerospace Conference*. Big Sky, MT, USA: IEEE, 2007, pp. 1–9. ISBN: 978-1-4244-0524-4. DOI: 10.1109/AERO.2007.352699. URL: http://ieeexplore.ieee.org/document/4161563/ (visited on 03/08/2023).

[53] Michael C. Burl et al. “ROCKSTER: Onboard Rock Segmentation Through Edge Regrouping”. en. In: *Journal of Aerospace Information Systems* 13.8 (Aug. 2016), pp. 329–342. ISSN: 1940-3151, 2327-3097. DOI: 10.2514/1.I010381. URL: https://arc.aiaa.org/doi/10.2514/1.I010381 (visited on 03/21/2023).

[54] David A. Spencer et al. “Phoenix Landing Site Hazard Assessment and Selection”. en. In: *Journal of Spacecraft and Rockets* 46.6 (Nov. 2009), pp. 1196–1201. ISSN: 0022-4650, 1533-6794. DOI: 10.2514/1.43932. URL: https://arc.aiaa.org/doi/10.2514/1.43932 (visited on 03/21/2023).

[55] Heather Dunlop, David R. Thompson, and David Wettergreen. “Multi-scale Features for Detection and Segmentation of Rocks in Mars Images”. In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. ISSN: 1063-6919. June 2007, pp. 1–7. DOI: 10.1109/CVPR.2007.383257.

[56] Xueming Xiao et al. “A Kernel-Based Multi-Featured Rock Modeling and Detection Framework for a Mars Rover”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2021). Conference Name: IEEE Transactions on Neural Networks and Learning Systems, pp. 1–10. ISSN: 2162-2388. DOI: 10.1109/TNNLS.2021.3131206.

[57] Shijie Xiao et al. “Robust Kernel Low-Rank Representation”. In: *IEEE Transactions on Neural Networks and Learning Systems* 27.11 (Nov. 2016). Conference Name: IEEE Transactions on Neural Networks and Learning Systems, pp. 2268–2281. ISSN: 2162-2388. DOI: 10.1109/TNNLS.2015.2472284.

[58] Jesús Ariel Carrasco-Ochoa et al., eds. *Pattern Recognition: 11th Mexican Conference, MCPR 2019, Querétaro, Mexico, June 26–29, 2019, Proceedings*. en. Vol. 11524. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019. ISBN: 978-3-030-21076-2 978-3-030-21077-9. DOI: 10.1007/978-3-030-21077-9. URL: http://link.springer.com/10.1007/978-3-030-21077-9 (visited on 03/23/2023).

[59] Rikiya Yamashita et al. “Convolutional neural networks: an overview and application in radiology”. en. In: *Insights into Imaging* 9.4 (Aug. 2018), pp. 611–629. ISSN: 1869-4101. DOI: 10.1007/s13244-018-0639-9. URL: https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9 (visited on 12/28/2023).

[60] Georgios Petrakis and Panagiotis Partsinevelos. *Lunar Ground Segmentation Using a Modified U-Net Neural Network*. en. preprint. In Review, Sept. 2023. DOI: 10.21203/rs.3.rs-3363458/v1. URL: https://www.researchsquare.com/article/rs-3363458/v1 (visited on 01/12/2024).

[61] C C Bedford et al. “OVERVIEW AND INITIAL RESULTS OF SAND-E: SEMI-AUTONOMOUS NAVIGATION FOR DETRITAL ENVIRONMENTS”. en. In: *51st Lunar and Planetary Science Conference (2020)* (2020). URL: https://ntrs.nasa.gov/api/citations/20200001815/downloads/20200001815.pdf.

[62] K. R. Raimalwala et al. “Autonomous Soil Assessment System: Contextualizing Rocks, Anomalies, and Terrains in Exploratory Robotic Science (ASAS-CRATERS)”. In: 2241 (May 2020). Conference Name: Lunar Surface Science Workshop ADS Bibcode: 2020LPICo2241.5124R, p. 5124. URL: https://ui.adsabs.harvard.edu/abs/2020LPICo2241.5124R (visited on 12/25/2023).

[63] *Chang’e 3 Mission Gallery – Change*. en-US. URL: https://spaceflight101.com/change/change-3-mission-gallery/ (visited on 01/17/2024).

[64] Michael Banks. “Japanese private craft crash lands on Moon”. en. In: *Physics World* 36.6 (June 2023), pp. 11i–11i. ISSN: 0953-8585, 2058-7058. DOI: 10.1088/2058-7058/36/06/13. URL: https://iopscience.iop.org/article/10.1088/2058-7058/36/06/13 (visited on 12/25/2023).

[65] Sara Almaeeni, Sebastian Els, and Hamad Almarzooqi. “The Rashid rover: to guide the way for the next generation lunar missions and solar system exploration”. In: (Apr. 2021). Conference Name: EGU General Assembly Conference Abstracts ADS Bibcode: 2021EGUGA..2314732A, EGU21–14732. DOI: 10.5194/egusphere-egu21-14732. URL: https://ui.adsabs.harvard.edu/abs/2021EGUGA..2314732A (visited on 12/25/2023).

[66] Faisal Khan, Saqib Salahuddin, and Hossein Javidnia. “Deep Learning-Based Monocular Depth Estimation Methods—A State-of-the-Art Review”. en. In: *Sensors* 20.8 (Apr. 2020), p. 2272. ISSN: 1424-8220. DOI: 10.3390/s20082272. URL: https://www.mdpi.com/1424-8220/20/8/2272 (visited on 01/01/2024).

[67] David Warden. “Analysis of Passive Radiation Shielding from Galactic Cosmic Rays in Cis-Lunar Space”. en. PhD thesis. Rice University, Apr. 2019. URL: https://scholarship.rice.edu/handle/1911/105991.

[68] Alankrita Isha Mrigakshi et al. “Estimation of Galactic Cosmic Ray exposure inside and outside the Earth’s magnetosphere during the recent solar minimum between solar cycles 23 and 24”. en. In: *Advances in Space Research* 52.5 (Sept. 2013), pp. 979–987. ISSN: 02731177. DOI: 10.1016/j.asr.2013.05.007. URL: https://linkinghub.elsevier.com/retrieve/pii/S0273117713002627 (visited on 12/25/2023).

[69] James R. Schwank. *Basic Mechanisms of Radiation Effects in the Natural Space Radiation Environment*. 1994. URL: https://www.osti.gov/servlets/purl/10158182 (visited on 04/05/2023).

[70] Maci J. Harrell, G. Starr Schroeder, and Stephen A. Daire. “Lunar Environment, Overview”. en. In: *Handbook of Life Support Systems for Spacecraft and Extraterrestrial Habitats*. Ed. by Erik Seedhouse and David J Shayler. Cham: Springer International Publishing, 2021, pp. 1–23. ISBN: 978-3-319-09575-2. DOI: 10.1007/978-3-319-09575-2_15-1. URL: https://link.springer.com/10.1007/978-3-319-09575-2_15-1 (visited on 02/02/2023).

[71] Guenther Reitz, Thomas Berger, and Daniel Matthiae. “Radiation exposure in the moon environment”. en. In: *Planetary and Space Science* 74.1 (Dec. 2012), pp. 78–83. ISSN: 00320633. DOI: 10.1016/j.pss.2012.07.014. URL: https://linkinghub.elsevier.com/retrieve/pii/S0032063312002085 (visited on 03/24/2023).

[72] Maurizio Spurio. *Particles and Astrophysics: A Multi-Messenger Approach*. en. Astronomy and Astrophysics Library. Cham: Springer International Publishing, 2015. ISBN: 978-3-319-08050-5 978-3-319-08051-2. DOI: 10.1007/978-3-319-08051-2. URL: https://link.springer.com/10.1007/978-3-319-08051-2 (visited on 03/31/2023).

[73] *The Deadly Van Allen Belts?* en. URL: https://www.nasa.gov/sites/default/files/files/SMIII_Problem7.pdf (visited on 05/09/2023).

[74] Ryan O'Hare. *Chinese space agency makes moon lander images available to the public*. Section: Science. Feb. 2016. URL: https://www.dailymail.co.uk/sciencetech/article-3426677/Explore-moon-s-rocky-terrain-stunning-Chinese-space-agency-lifts-veil-secrecy-make-lunar-lander-images-available-public-time.html (visited on 03/31/2023).

[75] C Meyer. *Lunar Regolith*. en. 2003. URL: https://curator.jsc.nasa.gov/lunar/letss/regolith.pdf (visited on 03/31/2023).

[76] M. P. Golombek et al. "Rock size-frequency distributions on Mars and implications for Mars Exploration Rover landing safety and operations". en. In: *Journal of Geophysical Research: Planets* 108.E12 (2003). _eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2002JE002035. ISSN: 2156-2202. DOI: 10.1029/2002JE002035. URL: https://onlinelibrary.wiley.com/doi/abs/10.1029/2002JE002035 (visited on 12/06/2023).

[77] Joshua L. Bandfield et al. "Lunar surface rock abundance and regolith fines temperatures derived from LRO Diviner Radiometer data". en. In: *Journal of Geophysical Research: Planets* 116.E12 (2011). _eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2011JE003866. ISSN: 2156-2202. DOI: 10.1029/2011JE003866. URL: https://onlinelibrary.wiley.com/doi/abs/10.1029/2011JE003866 (visited on 12/25/2023).

[78] Ian A. Crawford. "Lunar resources: A review". en. In: *Progress in Physical Geography: Earth and Environment* 39.2 (Apr. 2015), pp. 137–167. ISSN: 0309-1333, 1477-0296. DOI: 10.1177/0309133314567585. URL: http://journals.sagepub.com/doi/10.1177/0309133314567585 (visited on 03/27/2023).

[79] M. B. Duke. "Development of the Moon". en. In: *Reviews in Mineralogy and Geochemistry* 60.1 (Jan. 2006), pp. 597–655. ISSN: 1529-6466. DOI: 10.2138/rmg.2006.60.6. URL: https://pubs.geoscienceworld.org/rimg/article/60/1/597-655/140785 (visited on 02/27/2023).

[80] Heather Quinn. "Radiation effects in reconfigurable FPGAs". en. In: *Semiconductor Science and Technology* 32.4 (Apr. 2017), p. 044001. ISSN: 0268-1242, 1361-6641. DOI: 10.1088/1361-6641/aa57f6. URL: https://iopscience.iop.org/article/10.1088/1361-6641/aa57f6 (visited on 04/06/2023).

[81] F.W. Sexton. "Destructive single-event effects in semiconductor devices and ICs". en. In: *IEEE Transactions on Nuclear Science* 50.3 (June 2003), pp. 603–621. ISSN: 0018-9499, 1558-1578. DOI: 10.1109/TNS.2003.813137. URL: https://ieeexplore.ieee.org/document/1208579/ (visited on 04/06/2023).

[82] Muhammad Fayyaz and Tanya Vladimirova. "Fault-Tolerant Distributed approach to satellite On-Board Computer design". en. In: *2014 IEEE Aerospace Conference*. Big Sky, MT, USA: IEEE, Mar. 2014, pp. 1–12. ISBN: 978-1-4799-1622-1 978-1-4799-5582-4. DOI: 10.1109/AERO.2014.6836199. URL: http://ieeexplore.ieee.org/document/6836199/ (visited on 02/16/2023).

[83] *Redundant Systems: Definition, Important, Working Method*. en-US. URL: https://dexonsystems.com/blog/redundant-systems (visited on 05/04/2023).

[84] Abel Paz and Antonio Plaza. "A new morphological anomaly detection algorithm for hyperspectral images and its GPU implementation". en. In: San Diego, California, USA, Sept. 2011, p. 815708. DOI: 10.1117/12.892282. URL: http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.892282 (visited on 12/27/2023).

[85] J Serrano. "Introduction to FPGA design". en. In: *CERN Accelerator School: Specialized Course on Digital Signal Processing* (2008), pp. 231–247.

[86] Gaetan Bahl et al. “Low-Power Neural Networks for Semantic Segmentation of Satellite Images”. en. In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. Seoul, Korea (South): IEEE, Oct. 2019, pp. 2469–2476. ISBN: 978-1-72815-023-9. DOI: 10.1109/ICCVW.2019.00302. URL: https://ieeexplore.ieee.org/document/9022273/ (visited on 06/30/2023).

[87] Zsolt Voroshazi et al. “An embedded CNN-UM Global Analogic Programming Unit implementation on FPGA”. en. In: *2006 10th International Workshop on Cellular Neural Networks and Their Applications*. Istanbul, Turkey: IEEE, Aug. 2006, pp. 1–5. ISBN: 978-1-4244-0639-5 978-1-4244-0640-1. DOI: 10.1109/CNNA.2006.341652. URL: http://ieeexplore.ieee.org/document/4145892/ (visited on 12/27/2023).

[88] M. Perko et al. “Low-cost, high-performance CNN simulator implemented in FPGA”. en. In: *Proceedings of the 2000 6th IEEE International Workshop on Cellular Neural Networks and their Applications (CNNA 2000) (Cat. No.00TH8509)*. Catania, Italy: IEEE, 2000, pp. 277–282. ISBN: 978-0-7803-6344-1. DOI: 10.1109/CNNA.2000.876858. URL: http://ieeexplore.ieee.org/document/876858/ (visited on 12/27/2023).

[89] Xuechao Wei et al. “Automated Systolic Array Architecture Synthesis for High Throughput CNN Inference on FPGAs”. en. In: *Proceedings of the 54th Annual Design Automation Conference 2017*. Austin TX USA: ACM, June 2017, pp. 1–6. ISBN: 978-1-4503-4927-7. DOI: 10.1145/3061639.3062207. URL: https://dl.acm.org/doi/10.1145/3061639.3062207 (visited on 12/27/2023).

[90] Huynh Vinh Phu et al. “Design and Implementation of Configurable Convolutional Neural Network on FPGA”. en. In: *2019 6th NAFOSTED Conference on Information and Computer Science (NICS)*. Hanoi, Vietnam: IEEE, Dec. 2019, pp. 298–302. ISBN: 978-1-72815-163-2. DOI: 10.1109/NICS48868.2019.9023810. URL: https://ieeexplore.ieee.org/document/9023810/ (visited on 12/27/2023).

[91] Ran Wu et al. “Accelerating Neural Network Inference on FPGA-Based Platforms—A Survey”. en. In: *Electronics* 10.9 (Apr. 2021), p. 1025. ISSN: 2079-9292. DOI: 10.3390/electronics10091025. URL: https://www.mdpi.com/2079-9292/10/9/1025 (visited on 12/28/2023).

[92] Norman P. Jouppi et al. “In-Datacenter Performance Analysis of a Tensor Processing Unit”. en. In: *Proceedings of the 44th Annual International Symposium on Computer Architecture*. Toronto ON Canada: ACM, June 2017, pp. 1–12. ISBN: 978-1-4503-4892-8. DOI: 10.1145/3079856.3080246. URL: https://dl.acm.org/doi/10.1145/3079856.3080246 (visited on 12/27/2023).

[93] Justin Goodwill et al. “NASA SpaceCube Edge TPU SmallSat Card for Autonomous Operations and Onboard Science -Data Analysis”. en. In: ().

[94] Gianluca Furano et al. “Towards the Use of Artificial Intelligence on the Edge in Space Systems: Challenges and Opportunities”. In: *IEEE Aerospace and Electronic Systems Magazine* 35.12 (Dec. 2020). Conference Name: IEEE Aerospace and Electronic Systems Magazine, pp. 44–56. ISSN: 1557-959X. DOI: 10.1109/MAES.2020.3008468.

[95] *New Movidius Myriad X VPU Packs a Custom Neural Compute Engine*. en. Aug. 2017. URL: https://www.extremetech.com/computing/254772-new-movidius-myriad-x-vpu-packs-custom-neural-compute-engine (visited on 12/27/2023).

[96] *What is PC104?* URL: https://www.rtd.com/PC104/ (visited on 12/16/2023).

[97] Shuanglong Liu et al. “Optimizing CNN-based Segmentation with Deeply Customized Convolutional and Deconvolutional Architectures on FPGA”. en. In: *ACM Transactions on Reconfigurable Technology and Systems* 11.3 (Sept. 2018), pp. 1–22. ISSN: 1936-7406, 1936-7414. DOI: 10.1145/3242900. URL: https://dl.acm.org/doi/10.1145/3242900 (visited on 06/30/2023).

[98] *COTS Components in Spacecraft Systems: Understanding the Risk*. PDF. 2014. URL: https://www.nasa.gov/wp-content/uploads/2015/05/cots.pdf (visited on 01/12/2024).

[99] Anatoly Zak. *Luna-Grunt*. May 2021. URL: https://www.russianspaceweb.com/luna_grunt.html (visited on 05/23/2023).

[100] https://www.jpl.nasa.gov. *CADRE*. en-US. URL: https://www.jpl.nasa.gov/missions/cadre (visited on 12/26/2023).

[101] *IBM RAD6000*. en. Page Version ID: 1153482631. May 2023. URL: https://en.wikipedia.org/w/index.php?title=IBM_RAD6000&oldid=1153482631 (visited on 05/23/2023).

[102] *Comparison of embedded computer systems on board the Mars rovers*. en. Page Version ID: 1138595945. Feb. 2023. URL: https://en.wikipedia.org/w/index.php?title=Comparison_of_embedded_computer_systems_on_board_the_Mars_rovers&oldid=1138595945 (visited on 05/22/2023).

[103] *RAD750*. en. Page Version ID: 1147093404. Mar. 2023. URL: https://en.wikipedia.org/w/index.php?title=RAD750&oldid=1147093404 (visited on 05/24/2023).

[104] Tony Ho Tran. *NASA's New Mars Rover Is Less Powerful Than Many Smartphones*. URL: https://futurism.com/the-byte/nasas-new-mars-rover-is-less-powerful-than-many-smartphones (visited on 05/24/2023).

[105] Raja Roy, Curba Lampert, and Minyoung Kim. *Ingenuity of NASA: Designing the paradigm-changing Martian helicopter*. en. URL: https://mackinstitute.wharton.upenn.edu/wp-content/uploads/2022/03/Roy-Raja-Lampert-Curba-and-Kim-Minyoung_Ingenuity-of-NASA.pdf.

[106] *Ingenuity Mars Helicopter - Landing Press Kit*. en. URL: https://www.jpl.nasa.gov/news/press_kits/mars_2020/download/ingenuity_landing_press_kit.pdf (visited on 05/24/2023).

[107] *Snapdragon 801 Processor | Qualcomm*. en. URL: https://www.qualcomm.com/products/mobile/snapdragon/smartphones/snapdragon-8-series-mobile-platforms/snapdragon-processors-801 (visited on 05/24/2023).

[108] *snapdragon 801 processor product brief*. en. 2014. URL: https://www.qualcomm.com/content/dam/qcomm-martech/dm-assets/documents/snapdragon_801_processor_product_brief_122.pdf (visited on 05/24/2023).

[109] *Qualcomm Announces RB5 Robotics Platform - A Powerful SBC*. URL: https://www.anandtech.com/show/15856/qualcomm-announces-rb5-robotics-platform-a-powerful-sbc (visited on 05/24/2023).

[110] Emily Dunkel et al. "Benchmarking Deep Learning Inference of Remote Sensing Imagery on the Qualcomm Snapdragon And Intel Movidius Myriad X Processors Onboard the International Space Station". en. In: *IGARSS 2022 - 2022 IEEE International Geoscience and Remote Sensing Symposium*. Kuala Lumpur, Malaysia: IEEE, July 2022, pp. 5301–5304. ISBN: 978-1-66542-792-0. DOI: 10.1109/IGARSS46834.2022.9884906. URL: https://ieeexplore.ieee.org/document/9884906/ (visited on 05/22/2023).

[111] *snapdragon_801_processor_product_brief_122.pdf*. URL: https://www.qualcomm.com/content/dam/qcomm-martech/dm-assets/documents/snapdragon_801_processor_product_brief_122.pdf (visited on 12/26/2023).

[112] Kaizad Raimalwala et al. "ENABLING AUTONOMY IN COMMERCIAL-CLASS LUNAR MISSIONS". en. In: (Oct. 2020).

[113] Alan D. George and Christopher M. Wilson. "Onboard Processing With Hybrid and Reconfigurable Computing on Small Satellites". In: *Proceedings of the IEEE* 106.3 (Mar. 2018). Conference Name: Proceedings of the IEEE, pp. 458–470. ISSN: 1558-2256. DOI: 10.1109/JPROC.2018.2802438. URL: https://ieeexplore-ieee-org.tudelft.idm.oclc.org/document/8303008.

[114] L. Dadda. "The evolution of computer architectures". en. In: *[1991] Proceedings, Advanced Computer Technology, Reliable Systems and Applications*. Bologna, Italy: IEEE Comput. Soc. Press, 1991, pp. 9–16. ISBN: 978-0-8186-2141-3. DOI: 10.1109/CMPEUR.1991.257347. URL: http://ieeexplore.ieee.org/document/257347/ (visited on 12/22/2023).

[115] R. Duncan. "A survey of parallel computer architectures". en. In: *Computer* 23.2 (Feb. 1990), pp. 5–16. ISSN: 0018-9162, 1558-0814. DOI: 10.1109/2.44900. URL: https://ieeexplore.ieee.org/document/44900/ (visited on 12/22/2023).

[116] I.I. Arikpo, F.U. Ogban, and I.E. Eteng. "Von Neumann Architecture and Modern Computers". In: *Global Journal of Mathematical Sciennces* 6.2 (2007), pp. 97–103.

[117] Danijela Markovic et al. *Physics for Neuromorphic Computing*. arXiv:2003.04711 [physics]. Mar. 2020. URL: http://arxiv.org/abs/2003.04711 (visited on 12/22/2023).

[118] Andrew Steane. “Quantum computing”. In: (Aug. 1997). URL: https://iopscience.iop.org/article/10.1088/0034-4885/61/2/002/pdf (visited on 12/22/2023).

[119] *Moon exploration rovers from Lunar Zebro - Robohub*. URL: https://robohub.org/moon-exploration-rovers-from-lunar-zebro-what-if-you-would-use-this/ (visited on 12/13/2023).

[120] Manuele Brambilla et al. “Swarm robotics: a review from the swarm engineering perspective”. en. In: *Swarm Intelligence* 7.1 (Mar. 2013), pp. 1–41. ISSN: 1935-3812, 1935-3820. DOI: 10.1007/s11721-012-0075-2. URL: http://link.springer.com/10.1007/s11721-012-0075-2 (visited on 05/01/2023).

[121] Erol Şahin. “Swarm Robotics: From Sources of Inspiration to Domains of Application”. en. In: *Swarm Robotics*. Ed. by David Hutchison et al. Vol. 3342. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 10–20. ISBN: 978-3-540-24296-3 978-3-540-30552-1. DOI: 10.1007/978-3-540-30552-1_2. URL: http://link.springer.com/10.1007/978-3-540-30552-1_2 (visited on 05/01/2023).

[122] Heiko Hamann. *Swarm Robotics: A Formal Approach*. en. Cham: Springer International Publishing, 2018. ISBN: 978-3-319-74526-8 978-3-319-74528-2. DOI: 10.1007/978-3-319-74528-2. URL: http://link.springer.com/10.1007/978-3-319-74528-2 (visited on 05/01/2023).

[123] S.A. Curtis et al. “ANTS for Human Exploration and Development of Space”. en. In: *2003 IEEE Aerospace Conference Proceedings (Cat. No.03TH8652)*. Vol. 1. Big Sky, MT, USA: IEEE, 2003, pp. 1–261. ISBN: 978-0-7803-7651-9. DOI: 10.1109/AERO.2003.1235057. URL: http://ieeexplore.ieee.org/document/1235057/ (visited on 12/27/2023).

[124] https://www.jpl.nasa.gov. *CADRE*. en-US. URL: https://www.jpl.nasa.gov/missions/cadre (visited on 12/27/2023).

[125] Heiko Hamann and Heinz Wörn. “A framework of space–time continuous models for algorithm design in swarm robotics”. en. In: *Swarm Intelligence* 2.2-4 (Dec. 2008), pp. 209–239. ISSN: 1935-3812, 1935-3820. DOI: 10.1007/s11721-008-0015-3. URL: http://link.springer.com/10.1007/s11721-008-0015-3 (visited on 12/27/2023).

[126] Abdelhak Chatty et al. “Emergent complex behaviors for swarm robotic systems by local rules”. en. In: *2011 IEEE Workshop on Robotic Intelligence In Informationally Structured Space*. Paris, France: IEEE, Apr. 2011, pp. 69–76. ISBN: 978-1-4244-9885-7. DOI: 10.1109/RIISS.2011.5945791. URL: http://ieeexplore.ieee.org/document/5945791/ (visited on 12/27/2023).

[127] Mike Mannion and Barry Keepence. “SMART requirements”. en. In: *ACM SIGSOFT Software Engineering Notes* 20.2 (Apr. 1995), pp. 42–47. ISSN: 0163-5948. DOI: 10.1145/224155.224157. URL: https://dl.acm.org/doi/10.1145/224155.224157 (visited on 01/10/2024).

[128] Kevin Forsberg and Harold Mooz. “The Relationship of System Engineering to the Project Cycle”. en. In: *INCOSE International Symposium* 1.1 (Oct. 1991), pp. 57–65. ISSN: 2334-5837, 2334-5837. DOI: 10.1002/j.2334-5837.1991.tb01484.x. URL: https://incose.onlinelibrary.wiley.com/doi/10.1002/j.2334-5837.1991.tb01484.x (visited on 01/21/2024).

[129] David Walden et al. *Systems Engineering Handbook - a guide for system life cycle processes and activities*. 4th ed. Vol. 4. Wiley. URL: https://img1.wsimg.com/blobby/go/a430a7ae-a333-4c88-af76-fd5624bfbddd/downloads/INCOSE%20Systems%20Engineering%20Handbook%204e%202015%2007.pdf?ver=1604878104477 (visited on 12/20/2023).

[130] *verification | European Cooperation for Space Standardization*. URL: https://ecss.nl/item/?glossary_id=2690 (visited on 11/15/2023).

[131] Alan Boyle. *Elon Musk geeks out anew over his Mars plan*. Oct. 2016. URL: https://www.geekwire.com/2016/spacex-elon-musk-geeks-out-mars-reddit/ (visited on 06/02/2023).

[132] Bin Yu et al. “Reliability Evaluation and In-Orbit Residual Life Prediction for Satellite Lithium-Ion Batteries”. en. In: *Mathematical Problems in Engineering* 2018 (Dec. 2018), pp. 1–12. ISSN: 1024-123X, 1563-5147. DOI: 10.1155/2018/5918068. URL: https://www.hindawi.com/journals/mpe/2018/5918068/ (visited on 01/02/2024).

[133] Mengu Cho et al. “Spacecraft Charging Analysis of Large GEO Satellites Using MUSCAT”. en. In: *IEEE Transactions on Plasma Science* 40.4 (Apr. 2012), pp. 1248–1256. ISSN: 0093-3813, 1939-9375. DOI: 10.1109/TPS.2012.2187074. URL: http://ieeexplore.ieee.org/document/6156792/ (visited on 01/02/2024).

[134] *Acoustic Noise Requirement*. Jan. 1999. URL: https://llis.nasa.gov/lesson/787 (visited on 06/02/2023).

[135] *Zigbee Radios on the Ingenuity Mars Helicopter*. Nov. 2021. URL: https://soldersmoke.blogspot.com/2021/11/zigbee-radios-on-ingenuity-mars.html (visited on 12/20/2023).

[136] Manfred ”Dutch” Von Ehrenfried. *Perseverance and the Mars 2020 Mission: Follow the Science to Jezero Crater*. en. Cham: Springer International Publishing, 2022. ISBN: 978-3-030-92117-0 978-3-030-92118-7. DOI: 10.1007/978-3-030-92118-7. URL: https://link.springer.com/10.1007/978-3-030-92118-7 (visited on 11/06/2023).

[137] *Technology by the Zigbee Alliance enables communication on Mars*. en. URL: https://www.develcoproducts.com/blog/technology-by-the-zigbee-alliance-enables-communication-on-mars/ (visited on 11/06/2023).

[138] Thomas Hermelink et al. *Project Document of the Motherboard Design for Lunar Zebro*. en. Apr. 2020.

[139] Leonidas Kosmidis et al. “GPU4S: Embedded GPUs in Space”. en. In: *2019 22nd Euromicro Conference on Digital System Design (DSD)*. Kallithea, Greece: IEEE, Aug. 2019, pp. 399–405. ISBN: 978-1-72812-862-7. DOI: 10.1109/DSD.2019.00064. URL: https://ieeexplore.ieee.org/document/8875115/ (visited on 12/11/2023).

[140] Yunxiang Hu, Yuhao Liu, and Zhuovuan Liu. “A Survey on Convolutional Neural Network Accelerators: GPU, FPGA and ASIC”. en. In: *2022 14th International Conference on Computer Research and Development (ICCRD)*. Shenzhen, China: IEEE, Jan. 2022, pp. 100–107. ISBN: 978-1-72817-721-2. DOI: 10.1109/ICCRD54409.2022.9730377. URL: https://ieeexplore.ieee.org/document/9730377/ (visited on 12/11/2023).

[141] Vasileios Leon et al. “FPGA & VPU Co-Processing in Space Applications: Development and Testing with DSP/AI Benchmarks”. In: *2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*. Nov. 2021, pp. 1–5. DOI: 10.1109/ICECS53924.2021.9665462.

[142] Vasileios Leon et al. “FPGA & VPU Co-Processing in Space Applications: Development and Testing with DSP/AI Benchmarks”. In: *2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*. Nov. 2021, pp. 1–5. DOI: 10.1109/ICECS53924.2021.9665462.

[143] Federico Furlán et al. “CNN Based Detectors on Planetary Environments: A Performance Evaluation”. en. In: *Frontiers in Neurorobotics* 14 (Oct. 2020), p. 590371. ISSN: 1662-5218. DOI: 10.3389/fnbot.2020.590371. URL: https://www.frontiersin.org/articles/10.3389/fnbot.2020.590371/full (visited on 07/03/2023).

[144] Erik Ostrowski, Stefan Kaufmann, and IT-Designers GmbH. “Survey of alternative hardware for Neural Network computation in the context of Computer Vision”. en. In: (May 2020). URL: https://www.researchgate.net/publication/341495230.

[145] *TensorFlow models on the Edge TPU*. en-us. URL: https://coral.ai/docs/edgetpu/models-intro/#compatibility-overview (visited on 12/28/2023).

[146] *Ubotica CogniSat XE2 Edge Compute Payload Processor*. en. URL: https://ubotica.com/wp-content/uploads/2023/11/Ubotica-CogniSatXE2-On-Board-AI-Payload-Processor-Brief-v0.5.pdf (visited on 10/18/2023).

[147] *Weight & Count Chart of Popular Fasteners*. en. URL: https://itafasteners.com/weight-chart.php (visited on 01/15/2024).

[148] Aubrey Dunne. *CogniSAT-XE2 Interface Control Document*. en. Aug. 2023.

[149] Eben Upton. *Raspberry Pi 4 on sale now from $35*. en-GB. June 2019. URL: https://www.raspberrypi.com/news/raspberry-pi-4-on-sale-now-from-35/ (visited on 12/30/2023).

[150] Hesam Izakian and Witold Pedrycz. “Anomaly detection in time series data using a fuzzy c-means clustering”. en. In: *2013 Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS)*. Edmonton, AB, Canada: IEEE, June 2013, pp. 1513–1518. ISBN: 978-1-4799-0348-1. DOI: 10.1109/IFSA-NAFIPS.2013.6608627. URL: http://ieeexplore.ieee.org/document/6608627/ (visited on 12/30/2023).

[151] Kyle Hundman et al. “Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding”. en. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. London United Kingdom: ACM, July 2018, pp. 387–395. ISBN: 978-1-4503-5552-0. DOI: 10.1145/3219819.3219845. URL: https://dl.acm.org/doi/10.1145/3219819.3219845 (visited on 12/30/2023).

[152] Yangdong He and Jiabao Zhao. “Temporal Convolutional Networks for Anomaly Detection in Time Series”. en. In: *Journal of Physics: Conference Series* 1213.4 (June 2019), p. 042050. ISSN: 1742-6588, 1742-6596. DOI: 10.1088/1742-6596/1213/4/042050. URL: https://iopscience.iop.org/article/10.1088/1742-6596/1213/4/042050 (visited on 12/30/2023).

[153] T. Watteyne. “Lower-power wireless mesh networks for machine-to-machine communications using the IEEE802.15.4 standard”. en. In: *Machine-to-machine (M2M) Communications*. Elsevier, 2015, pp. 63–77. ISBN: 978-1-78242-102-3. DOI: 10.1016/B978-1-78242-102-3.00004-6. URL: https://linkinghub.elsevier.com/retrieve/pii/B9781782421023000046 (visited on 12/31/2023).

[154] *2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver*. en. URL: https://www.ti.com/lit/ds/symlink/cc2420.pdf (visited on 01/11/2023).

[155] Ugo Colesanti and Silvia Santini. “The Collection Tree Protocol for the Castalia Wireless Sensor Networks Simulator”. en. In: (July 2011). URL: https://www.researchgate.net/publication/228429268_The_Collection_Tree_Protocol_for_the_Castalia_Wireless_Sensor_Networks_Simulator.

[156] “ZigBee and IEEE 802.15.4 Protocol Layers”. In: *ZigBee Wireless Networks and Transceivers*. newnespress, Dec. 2008, pp. 33–135. ISBN: 10.1016/B978-0-7506-8393-7.00003-0. URL: https://www.researchgate.net/publication/288216311_ZigBee_and_IEEE_802154_Protocol_Layers (visited on 12/12/2023).

[157] *IEEE 802.15.4*. en. Page Version ID: 1188052116. Dec. 2023. URL: https://en.wikipedia.org/w/index.php?title=IEEE_802.15.4&oldid=1188052116 (visited on 12/31/2023).

[158] Mario Coppola. *Automatic Design of Verifiable Robot Swarms*. 2021.

[159] V. Gazi and K.M. Passino. “A class of attraction/repulsion functions for stable swarm aggregations”. en. In: *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.* Vol. 3. Las Vegas, NV, USA: IEEE, 2002, pp. 2842–2847. ISBN: 978-0-7803-7516-1. DOI: 10.1109/CDC.2002.1184277. URL: http://ieeexplore.ieee.org/document/1184277/ (visited on 01/01/2024).

[160] Dong Hun Kim, Hua Wang, and Seiichi Shin. “Decentralized Control of Autonomous Swarm Systems Using Artificial Potential Functions: Analytical Design Guidelines”. en. In: *Journal of Intelligent and Robotic Systems* 45.4 (Apr. 2006), pp. 369–394. ISSN: 0921-0296, 1573-0409. DOI: 10.1007/s10846-006-9050-8. URL: http://link.springer.com/10.1007/s10846-006-9050-8 (visited on 06/21/2023).

[161] Nathalie Majcherczyk et al. *Decentralized Connectivity-Preserving Deployment of Large-Scale Robot Swarms*. en. arXiv:1806.00150 [cs]. May 2018. URL: http://arxiv.org/abs/1806.00150 (visited on 01/07/2024).

[162] Xiaohai Li, Jizong Xiao, and Zhijun Cai. “Stable Flocking of Swarms Using Local Information”. en. In: *2005 IEEE International Conference on Systems, Man and Cybernetics*. Vol. 4. Waikoloa, HI, USA: IEEE, 2005, pp. 3921–3926. ISBN: 978-0-7803-9298-4. DOI: 10.1109/ICSMC.2005.1571758. URL: http://ieeexplore.ieee.org/document/1571758/ (visited on 01/07/2024).

[163] Craig W. Reynolds. “Flocks, herds and schools: A distributed behavioral model”. en. In: *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. ACM, Aug. 1987, pp. 25–34. ISBN: 978-0-89791-227-3. DOI: 10.1145/37401.37406. URL: https://dl.acm.org/doi/10.1145/37401.37406 (visited on 06/08/2023).

[164] B Quaghebeur, D. Heyndrickx, and J Wera. “ESA’s Space ENVironment Information System (SPENVIS): a Web-Based Tool for Assessing Radiation Doses and Effects in Spacecraft Systems”. en. In: *San Diego* (2005). URL: https://orfeo.belnet.be/handle/internal/4602.

[165] Dabin Xue et al. “Forward-Looking Study of Solar Maximum Impact in 2025: Effects of Satellite Navigation Failure on Aviation Network Operation in the Greater Bay Area, China”. en. In: *Space Weather* 21.12 (2023). _eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2023SW003678, e2023SW003678. ISSN: 1542-7390. DOI: 10.1029/2023SW003678. URL: https://onlinelibrary.wiley.com/doi/abs/10.1029/2023SW003678 (visited on 01/08/2024).

[166] William F Dietrich, Margaret A Shea, and Don F Smart. “CREME96: A Revision of the Cosmic Ray Effects on Micro-Electronics Code”. en. In: *IEEE TRANSACTIOdS ON NUCLEAR SCIENCE* 44.6 (Dec. 1997). URL: https://ieeexplore.ieee.org/document/659030.

[167] Xapsos, O’Neill, and T. Paul O’Brien. “Near-Earth Space Radiation Models”. en. In: *IEEE Transactions on Nuclear Science* 60.3 (June 2013), pp. 1691–1705. ISSN: 0018-9499, 1558-1578. DOI: 10.1109/TNS.2012.2225846. URL: http://ieeexplore.ieee.org/document/6375793/ (visited on 01/08/2024).

[168] R.A. Nymmik. “Improved environment radiation models”. en. In: *Advances in Space Research* 40.3 (Jan. 2007), pp. 313–320. ISSN: 02731177. DOI: 10.1016/j.asr.2006.12.028. URL: https://linkinghub.elsevier.com/retrieve/pii/S0273117706007861 (visited on 01/08/2024).

[169] F Faccio. “Estimate of the single event upset (SEU) rate in CMS”. en. In: *4th Workshop on Electronics for LHC Experiments* (Sept. 1998). URL: https://cds.cern.ch/record/405088/files/LHCC-98-36_119.pdf.

[170] *ESA developing deep sub-micron technology to deliver smarter satellites*. en. URL: https://www.esa.int/Enabling_Support/Space_Engineering_Technology/ESA_developing_deep_sub-micron_technology_to_deliver_smarter_satellites (visited on 01/11/2024).

[171] Devin P. Ramaswami et al. “Single Event Upset Characterization of the Intel Movidius Myriad X VPU and Google Edge TPU Accelerators Using Proton Irradiation”. en. In: *2022 IEEE Radiation Effects Data Workshop (REDW) (in conjunction with 2022 NSREC)*. Provo, UT, USA: IEEE, July 2022, pp. 1–3. ISBN: 978-1-66548-856-3. DOI: 10.1109/REDW56037.2022.9921608. URL: https://ieeexplore.ieee.org/document/9921608/ (visited on 12/12/2023).

[172] David M. Hiemstra and Valeri Kirischian. “Single Event Upset Characterization of the Zynq-7000 ARM® Cortex™-A9 Processor Unit Using Proton Irradiation”. en. In: *2015 IEEE Radiation Effects Data Workshop (REDW)*. Boston, MA, USA: IEEE, July 2015, pp. 1–3. ISBN: 978-1-4673-7641-9. DOI: 10.1109/REDW.2015.7336735. URL: http://ieeexplore.ieee.org/document/7336735/ (visited on 01/09/2024).

[173] Leénie Buckley et al. “Radiation Test and in Orbit Performance of MpSoC AI Accelerator”. In: *2022 IEEE Aerospace Conference (AERO)*. ISSN: 1095-323X. Mar. 2022, pp. 1–9. DOI: 10.1109/AERO53065.2022.9843440.

[174] Xuecheng Du et al. “Analysis of sensitive blocks of soft errors in the Xilinx Zynq-7000 System-on-Chip”. en. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 940 (Oct. 2019), pp. 125–128. ISSN: 01689002. DOI: 10.1016/j.nima.2019.06.015. URL: https://linkinghub.elsevier.com/retrieve/pii/S0168900219308460 (visited on 01/08/2024).

[175] Xiuhai Cui et al. “Mitigating single event upset method for Zynq MPSoC”. en. In: *Microelectronics Reliability* 100-101 (Sept. 2019), p. 113384. ISSN: 00262714. DOI: 10.1016/j.microrel.2019.06.076. URL: https://linkinghub.elsevier.com/retrieve/pii/S0026271419305116 (visited on 01/11/2024).

[176] M. Y. Hsiao. “A Class of Optimal Minimum Odd-weight-column SEC-DED Codes”. In: *IBM Journal of Research and Development* 14.4 (July 1970). Conference Name: IBM Journal of Research and Development, pp. 395–401. ISSN: 0018-8646. DOI: 10.1147/rd.144.0395. URL: https://ieeexplore.ieee.org/document/5391627 (visited on 01/13/2024).

[177] Dr Véronique Ferlet-Cavrois. *Overview of Radiation Test Activities on Memories at ESA*. en. ESA ESTEC, TEC-QEC, Apr. 2015. URL: https://www.esa.int/gsp/ACT/doc/EVENTS/memristor_workshop/ACT-NAN-0415_6_ferlet-cavrois_2015-04-30%20ResistiveMem%20workshop.pdf (visited on 01/01/2024).

[178] Alok Kumar Kasgar and Mukesh Kumar Dhariwal. "A Review Paper of Message Digest 5 (MD5)". en. In: *Management Research* 1.4 (2013). URL: http://www.ijmemr.org/Publication/V1I4/IJMEMR-V1I4-005.pdf.

[179] Joshua Lamorie and Francesco Ricci. "MicroSD Operational Experience and Fault-Mitigation Techniques". en. In: *Conference on Small Satellites, 29th Annual AIAA/USU* (2015). URL: https://digitalcommons.usu.edu/smallsat/2015/all2015/70/.

[180] George Lentaris et al. "TID Evaluation System With On-Chip Electron Source and Programmable Sensing Mechanisms on FPGA". en. In: *IEEE Transactions on Nuclear Science* 66.1 (Jan. 2019), pp. 312–319. ISSN: 0018-9499, 1558-1578. DOI: 10.1109/TNS.2018.2885713. URL: https://ieeexplore.ieee.org/document/8570836/ (visited on 12/13/2023).

[181] *LUNAR RADIOS – CONNECTING SURFACE ASSETS WITH LINE OF SIGHT*. en-US. URL: https://missioncontrolspace.com/stories/lunar-radios-connecting-surface-assets-with-line-of-sight/ (visited on 01/17/2024).

[182] Mahmoud Mostapha. "A NOVEL DIFFUSION TENSOR IMAGING-BASED COMPUTER-AIDED DIAGNOSTIC SYSTEM FOR EARLY DIAGNOSIS OF AUTISM". PhD thesis. Aug. 2014. DOI: 10.13140/2.1.2236.1283.

[183] Meibao Yao. *MarsData-V2, a rock segmentation dataset of real Martian scenes*. en. Sept. 2022. URL: https://ieee-dataport.org/documents/marsdata-v2-rock-segmentation-dataset-real-martian-scenes (visited on 11/20/2023).

[184] Romain Pessia and Quentin Jodelet. *Artificial Lunar Landscape Dataset*. en. Nov. 2023. URL: https://www.kaggle.com/datasets/romainpessia/artificial-lunar-rocky-landscape-dataset (visited on 11/20/2023).

[185] *CVIR-Lab/MarsData*. original-date: 2021-10-05T06:57:17Z. May 2023. URL: https://github.com/CVIR-Lab/MarsData (visited on 07/14/2023).

[186] Haiqiang Liu et al. "RockFormer: A U-Shaped Transformer Network for Martian Rock Segmentation". en. In: *IEEE Transactions on Geoscience and Remote Sensing* 61 (2023), pp. 1–16. ISSN: 0196-2892, 1558-0644. DOI: 10.1109/TGRS.2023.3235525. URL: https://ieeexplore.ieee.org/document/10012398/ (visited on 01/17/2024).

[187] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. en. arXiv:1505.04597 [cs]. May 2015. URL: http://arxiv.org/abs/1505.04597 (visited on 01/09/2024).

[188] Federico Furlán et al. "Rock Detection in a Mars-Like Environment Using a CNN". en. In: *Pattern Recognition*. Ed. by Jesús Ariel Carrasco-Ochoa et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pp. 149–158. ISBN: 978-3-030-21077-9. DOI: 10.1007/978-3-030-21077-9_14.

[189] Boyu Kuang et al. "Rock Segmentation in the Navigation Vision of the Planetary Rovers". en. In: *Mathematics* 9.23 (Nov. 2021), p. 3048. ISSN: 2227-7390. DOI: 10.3390/math9233048. URL: https://www.mdpi.com/2227-7390/9/23/3048 (visited on 01/09/2024).

[190] Zongwei Zhou et al. "UNet++: A Nested U-Net Architecture for Medical Image Segmentation". en. In: *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*. Ed. by Danail Stoyanov et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 3–11. ISBN: 978-3-030-00889-5. DOI: 10.1007/978-3-030-00889-5_1.

[191] Huimin Huang et al. "UNet 3+: A Full-Scale Connected UNet for Medical Image Segmentation". In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ISSN: 2379-190X. May 2020, pp. 1055–1059. DOI: 10.1109/ICASSP40776.2020.9053405. URL: https://ieeexplore.ieee.org/document/9053405?reason=concurrency (visited on 01/09/2024).

[192] *Artificial Lunar Landscape Dataset*. en. URL: https://www.kaggle.com/datasets/romainpessia/artificial-lunar-rocky-landscape-dataset (visited on 12/29/2023).

[193] Shreyas Fadnavis. “Image Interpolation Techniques in Digital Image Processing: An Overview”. In: *International Journal Of Engineering Research and Application* 4 (Nov. 2014), pp. 2248–962270. URL: https://ijera.com/papers/Vol4_issue10/Part%20-%201/K41007073.pdf.

[194] Haiqiang Liu et al. “RockFormer: A U-Shaped Transformer Network for Martian Rock Segmentation”. en. In: *IEEE Transactions on Geoscience and Remote Sensing* 61 (2023), pp. 1–16. ISSN: 0196-2892, 1558-0644. DOI: 10.1109/TGRS.2023.3235525. URL: https://ieeexplore.ieee.org/document/10012398/ (visited on 01/17/2024).

[195] Paulius Micikevicius et al. *Mixed Precision Training*. arXiv:1710.03740 [cs, stat]. Feb. 2018. URL: http://arxiv.org/abs/1710.03740 (visited on 12/10/2023).

[196] *OpenVINO 2023.2 — OpenVINO™ documentation — Version(2023.2)*. URL: https://docs.openvino.ai/2023.2/home.html (visited on 12/29/2023).

[197] *Setting Input Shapes — OpenVINO™ documentationCopy to clipboardCopy to clipboardCopy to clipboardCopy to clipboardCopy to clipboardCopy to clipboardCopy to clipboardCopy to clipboardCopy to clipboard — Version(2023.0)*. URL: https://docs.openvino.ai/2023.0/openvino_docs_MO_DG_prepare_model_convert_model_Converting_Model.html (visited on 12/29/2023).

[198] *Converting an ONNX Model — OpenVINO™ documentationCopy to clipboardCopy to clipboard — Version(2023.0)*. URL: https://docs.openvino.ai/2023.0/openvino_docs_MO_DG_prepare_model_convert_model_Convert_Model_From_ONNX.html (visited on 11/17/2023).

[199] *tf2onnx - Convert TensorFlow, Keras, Tensorflow.js and Tflite models to ONNX.* original-date: 2018-03-13T18:39:56Z. Nov. 2023. URL: https://github.com/onnx/tensorflow-onnx (visited on 11/17/2023).

[200] *What is a Container? | Docker*. en-US. URL: https://www.docker.com/resources/what-container/ (visited on 12/29/2023).

[201] *Raspberry Pi 2 Model B Desktop (Quad Core CPU 900MHz, 1GB RAM, Linux) : Amazon.nl: Electronics & Photo*. URL: https://www.amazon.nl/-/en/Raspberry-Model-Desktop-900MHz-Linux/dp/B00T2U7R7I?th=1 (visited on 11/21/2023).

[202] *A Comparison of ARM Cortex-A Series Processor Performance Classifications - Blog - Forlinx Embedded Technology Co., Ltd.* URL: https://www.forlinx.net/industrial-news/arm-cortex-a-series-processor-performance-344.html (visited on 12/29/2023).

[203] Suman Prathwani. *ARM cortex a7 vs ARM cortex a9 vs ARM cortex a15.* en. Dec. 2014. URL: https://israniankit.wordpress.com/2014/12/24/arm-cortex-a7-vs-arm-cortex-a9-vs-arm-cortex-a15/ (visited on 12/29/2023).

[204] *Power Consumption Benchmarks | Raspberry Pi Dramble*. URL: https://www.pidramble.com/wiki/benchmarks/power-consumption (visited on 12/04/2023).

[205] *Siglent SPD1305X 0 to 30V / 0 to 5Amp Single output power supply | siglent.eu*. en-GB. URL: https://www.siglent.eu/product/1140972/siglent-spd1305x-0-to-30v-0-to-5amp-single-output-power-supply (visited on 11/21/2023).

[206] *Intel® Ethernet Controller I210 Datasheet*. en. Jan. 2021. URL: https://cdrdv2-public.intel.com/333016/333016%20-%20I210_Datasheet_v_3_7.pdf.

[207] Xavier Berger. *XavierBerger/RPi-Monitor*. original-date: 2013-04-22T20:01:02Z. Jan. 2024. URL: https://github.com/XavierBerger/RPi-Monitor (visited on 01/21/2024).

[208] *ISIS-On-board-Computer-Brochure-v2-compressed.pdf*. URL: https://www.isispace.nl/wp-content/uploads/2016/02/ISIS-On-board-Computer-Brochure-v2-compressed.pdf (visited on 01/15/2024).

[209] Andrew G. Howard et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. arXiv:1704.04861 [cs]. Apr. 2017. DOI: 10.48550/arXiv.1704.04861. URL: http://arxiv.org/abs/1704.04861 (visited on 01/17/2024).

[210] Mark Sandler et al. *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. arXiv:1801.04381 [cs]. Mar. 2019. DOI: 10.48550/arXiv.1801.04381. URL: http://arxiv.org/abs/1801.04381 (visited on 01/17/2024).

[211] Jie Hu et al. *Squeeze-and-Excitation Networks*. arXiv:1709.01507 [cs] version: 4. May 2019. DOI: 10.48550/arXiv.1709.01507. URL: http://arxiv.org/abs/1709.01507 (visited on 01/17/2024).

A

Nominal Power Test Results

The following graphs display the results of the nominal power tests. These were not directly included in the report.

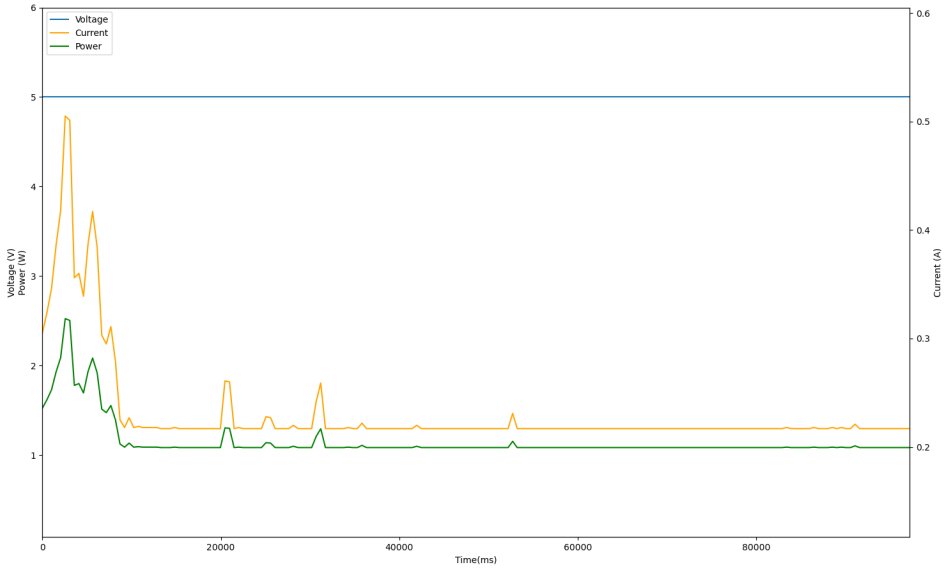


Figure A.1: Voltage, Current, and Power consumed by the RP2 in nominal conditions, when *no* mouse (USB), keyboard (USB) and display (HDMI) connected.

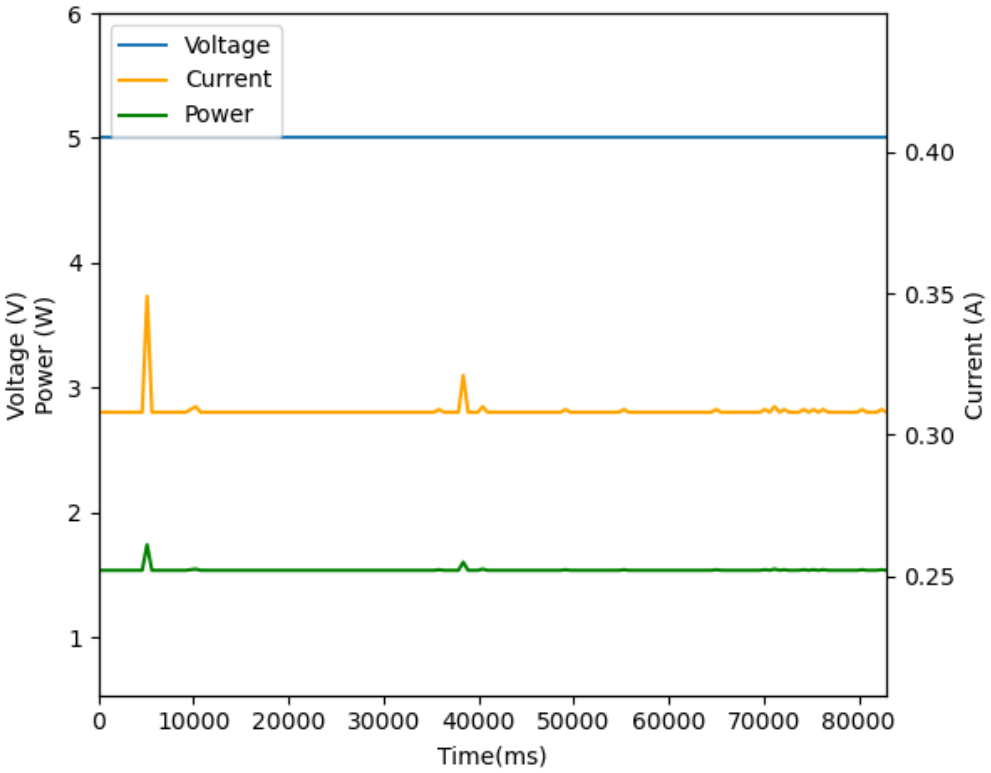


Figure A.2: Voltage, Current, and Power consumed by the RP2 in nominal conditions, when connected to a mouse (USB), keyboard (USB), and display (HDMI).

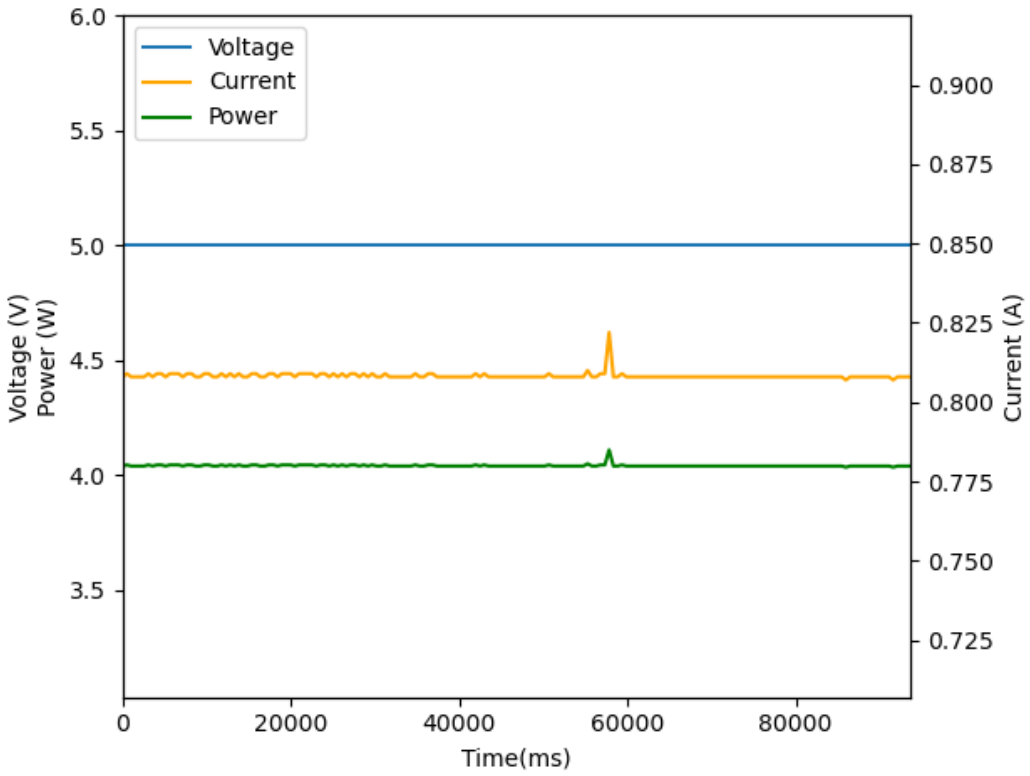


Figure A.3: Voltage, Current, and Power consumed by the RP2 and XE2 together in nominal conditions, when connected to a mouse (USB), keyboard (USB), and display (HDMI).

B

SPENVIS Settings

The following trajectory and radiation settings were used for the analysis in section 4.4.

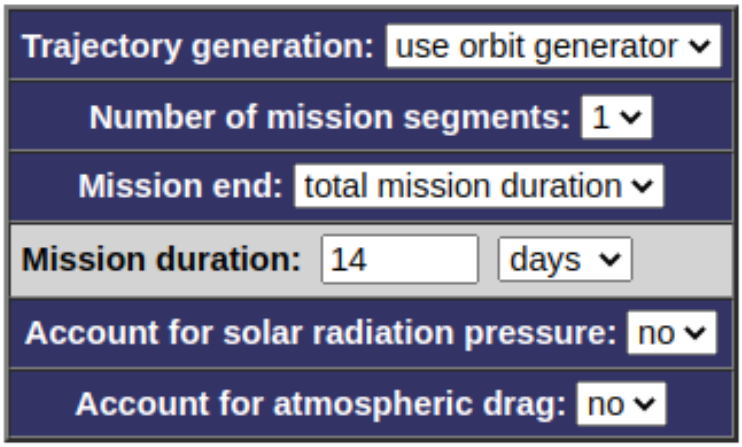


Figure B.1: Trajectory Settings

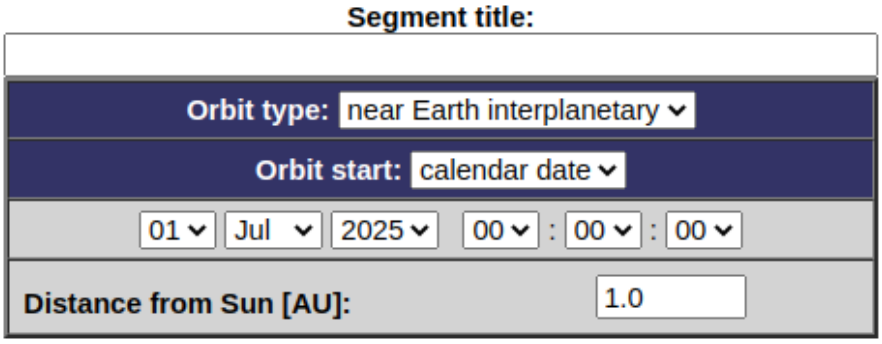


Figure B.2: Trajectory Settings

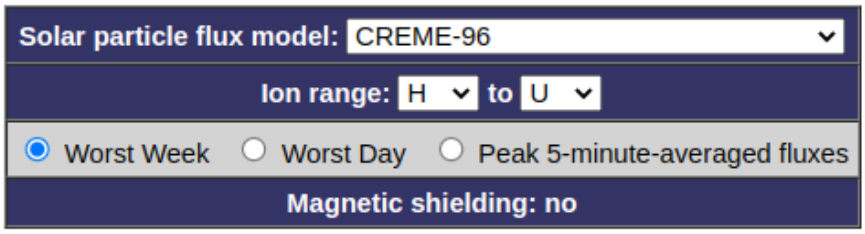


Figure B.3: Solar Particle Flux Model Settings

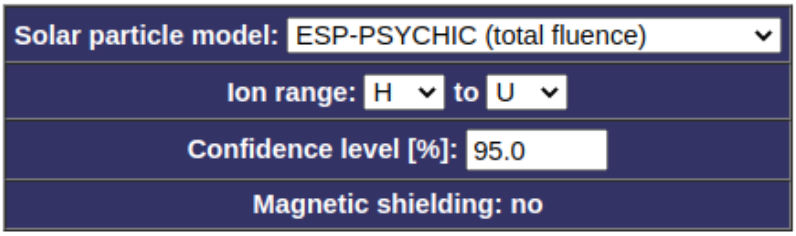


Figure B.4: Solar Particle Model Settings

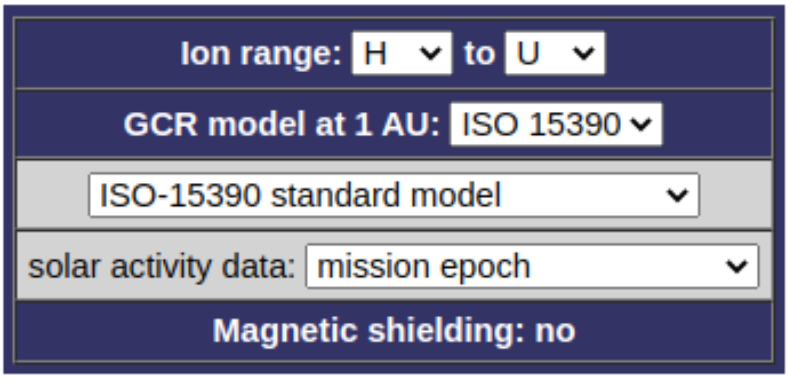


Figure B.5: GCR Model Settings

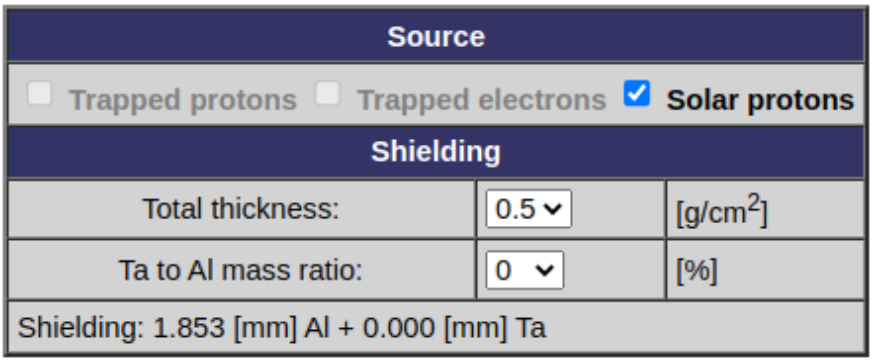


Figure B.6: Shielded Flux Settings

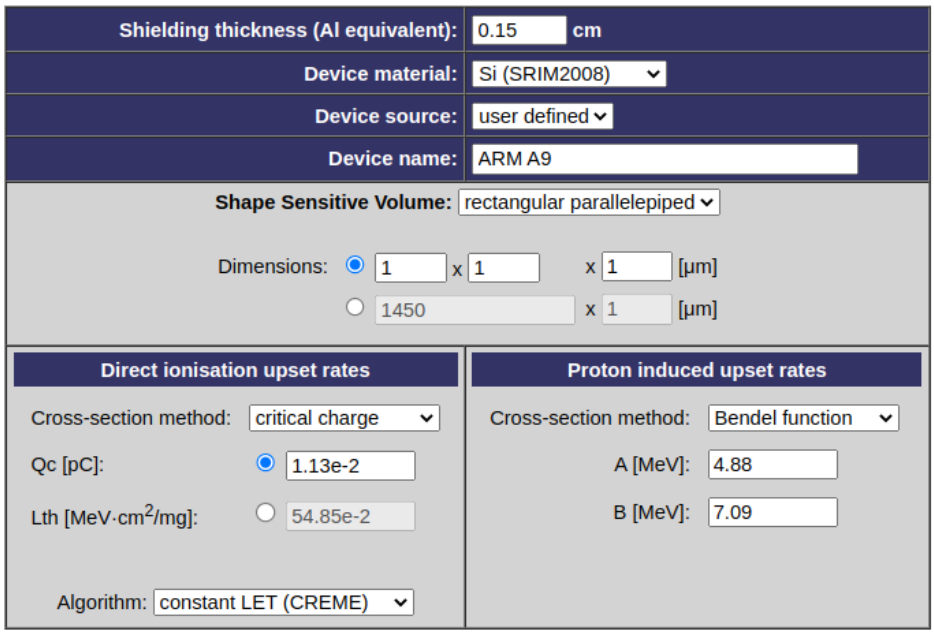


Figure B.7: Long-Term SEU Settings

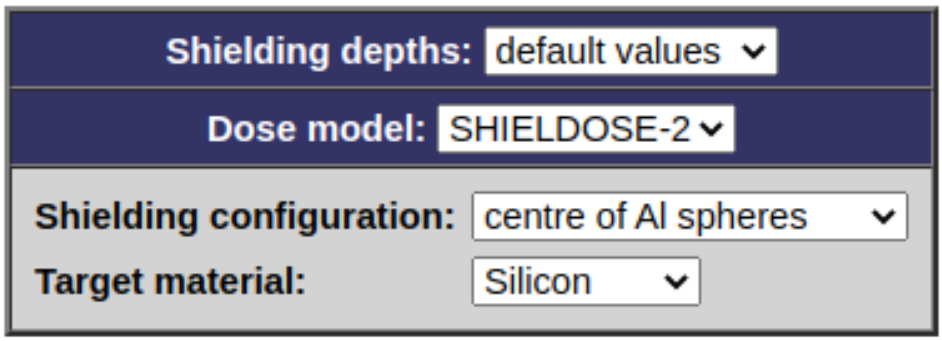


Figure B.8: Total Ionizing Dose Settings

C

Source Code

In this Appendix, all source code used in this project is found. Every piece of source code has a dedicated description attached.

C.1. Converting Lunar Artificial Dataset Masks from 3 to 2 classes

```
'''
Description: This script is dedicated to the conversion of the Lunar Artificial Dataset
    masks from 3 classes to 2 classes, as we are only interested in rock/non-rock.
Author: Thijs Bolscher (thijsbolscher18@live.nl)
Creation Date: August/September 2023
'''


from PIL import Image
import os
import numpy as np

def convert_masks(folder_path):
    output_folder = os.path.join(folder_path, 'converted_masks')
    os.makedirs(output_folder, exist_ok=True)  # Ensure output directory exists

    # Iterate through the images in the specified folder
    for filename in os.listdir(folder_path):
        if filename.endswith(".png"):  # Adjust file extension if needed
            img_path = os.path.join(folder_path, filename)

            # Extract the name without extension
            image_name, ext = os.path.splitext(filename)

            # Remove 'g_' prefix from filenames that match the pattern 'g_XY.png'
            if image_name.startswith('g_') and image_name[2:].isdigit():
                image_name = image_name[2:]

            # Open the image and convert it to a NumPy array
            img = Image.open(img_path)
            img_array = np.array(img)

            # Check if the image is grayscale and convert to 3 channels if necessary
            if len(img_array.shape) == 2:
                img_array = np.stack((img_array,) * 3, axis=-1)

            # Exclude the alpha channel from RGBA images
            if len(img_array.shape) == 3 and img_array.shape[2] == 4:
                img_array = img_array[:, :, :3]

            # Conversion logic for images
            if len(img_array.shape) == 3 and img_array.shape[2] == 3:
                # Define thresholds for identifying rock pixels
                blue_threshold, green_threshold, red_threshold = 130, 130, 150
```

```
                # Identify rock and not-rock pixels
                rock_mask = (img_array[:, :, 2] > blue_threshold) | (img_array[:, :, 1] > green_threshold)
                not_rock_mask = img_array[:, :, 0] > red_threshold

                # Apply mask to create a new binary image (rocks in white, rest in black)
                converted_mask = np.zeros_like(img_array)
                converted_mask[rock_mask] = [255, 255, 255]
                converted_mask[not_rock_mask] = [0, 0, 0]

                # Save the converted image
                converted_img = Image.fromarray(converted_mask.astype(np.uint8))
                converted_img.save(os.path.join(output_folder, f"{image_name}.png"))
                print(f"Converted {filename} and saved as {image_name}.png")
            else:
                print(f"Ignoring {filename}: not a valid RGB image")

    print('Masks converted and ready for training.')

# Replace 'folder_path' with the path to your folder containing masks
folder_path = '/path/to/your/mask/folder'
convert_masks(folder_path)
```

Source Code C.1: Converting Lunar Artificial Dataset Masks from 3 to 2 classes.

C.2. MultiResUNet Training

```
'''
Description: This script is dedicated to image segmentation using the MultiResUNet
    architecture, specifically tailored for Mars or Moon surface images. It encompasses
    data preprocessing, model construction, training, evaluation, and functions for model
    saving and loading. Emphasis is placed on handling imagery with efficient memory usage
    and performance optimization, crucial for space exploration imaging analysis.
Author: Thijs Bolscher (thijsbolscher18@live.nl)
Creation Date: August/September 2023
Based on MultiResUNet: https://github.com/nibtehaz/MultiResUNet
'''

import os
import cv2
import numpy as np
import gc
import re
import datetime
import tensorflow as tf
import keras
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from keras.models import Model, load_model
from keras.layers import Input, Conv2D, MaxPooling2D, Conv2DTranspose, concatenate,
    BatchNormalization, Activation, add
from keras.optimizers import Adam
from keras.callbacks import TensorBoard, ModelCheckpoint
from keras import backend as K

# Data preparation
img_files = sorted(next(os.walk('validation_images'))[2])
msk_files = sorted(next(os.walk('validation_masks'))[2])

X, Y, original_images = [], [], []
planar_data_all = []

for img_fl in tqdm(img_files):
    if img_fl.endswith('png'):
        img = cv2.imread(f'validation_images/{img_fl}', cv2.IMREAD_COLOR)
        resized_img = cv2.resize(img, (256, 192), interpolation=cv2.INTER_CUBIC).astype(np.
    float16)
```

```
        resized_original = cv2.resize(img, (256, 192), interpolation=cv2.INTER_CUBIC)

        planar_data_all.append(np.array(img))
        original_images.append(resized_original)

        X.append(resized_img)

        msk = cv2.imread(f'validation_masks/{img_fl}', cv2.IMREAD_GRAYSCALE)
        resized_msk = cv2.resize(msk, (256, 192), interpolation=cv2.INTER_CUBIC).astype(np.float16)
        Y.append(resized_msk)

X, Y = np.array(X), np.array(Y)
original_images = np.array(original_images)

original_images_train, original_images_test, Y_train, Y_test = train_test_split(original_images, Y, test_size=0.2, random_state=3)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=3)
Y_train, Y_test = Y_train.reshape((Y_train.shape[0], Y_train.shape[1], Y_train.shape[2], 1)), Y_test.reshape((Y_test.shape[0], Y_test.shape[1], Y_test.shape[2], 1))

original_images_train, original_images_test = original_images_train / 255, original_images_test / 255
X_train, X_test, Y_train, Y_test = X_train / 255, X_test / 255, Y_train / 255, Y_test / 255
Y_train, Y_test = np.round(Y_train, 0), np.round(Y_test, 0)

# 2D Convolutional layers
def conv2d_bn(x, filters, num_row, num_col, padding='same', strides=(1, 1), activation='relu', name=None):
    '''
    Apply a 2D Convolutional layer with batch normalization.
    Arguments:
        x: Input keras layer.
        filters: Number of filters.
        num_row: Number of rows in the filter.
        num_col: Number of columns in the filter.
        padding: Padding mode ('same' or 'valid').
        strides: Strides of the convolution.
        activation: Activation function to use.
        name: Name of the layer.
    Returns:
        Output keras layer after applying convolution and batch normalization.
    '''
    x = Conv2D(filters, (num_row, num_col), strides=strides, padding=padding, use_bias=False)(x)
    x = BatchNormalization(axis=3, scale=False)(x)
    if activation is not None:
        x = Activation(activation, name=name)(x)
    return x

# 2D Transposed Convolutional layers
def trans_conv2d_bn(x, filters, num_row, num_col, padding='same', strides=(2, 2), name=None):
    '''
    Apply a 2D Transposed Convolutional layer with batch normalization.
    Arguments:
        x: Input keras layer.
        filters: Number of filters.
        num_row: Number of rows in the filter.
        num_col: Number of columns in the filter.
        padding: Padding mode ('same' or 'valid').
        strides: Strides of the transposed convolution.
        name: Name of the layer.
    Returns:
        Output keras layer after applying transposed convolution and batch normalization.
    '''
    x = Conv2DTranspose(filters, (num_row, num_col), strides=strides, padding=padding)(x)
    x = BatchNormalization(axis=3, scale=False)(x)
    return x

# MultiRes Block
```

```
def MultiResBlock(U, inp, alpha=1.67):
    '''
    MultiRes Block as defined in the MultiResUNet architecture.
    Arguments:
        U: Number of filters in a corresponding UNet stage.
        inp: Input keras layer.
        alpha: Scaling factor for number of filters.
    Returns:
        Output keras layer after applying MultiRes Block.
    '''
    W = alpha * U
    shortcut = inp
    shortcut = conv2d_bn(shortcut, int(W*0.167) + int(W*0.333) + int(W*0.5), 1, 1,
    activation=None, padding='same')

    conv3x3 = conv2d_bn(inp, int(W*0.167), 3, 3, activation='relu', padding='same')
    conv5x5 = conv2d_bn(conv3x3, int(W*0.333), 3, 3, activation='relu', padding='same')
    conv7x7 = conv2d_bn(conv5x5, int(W*0.5), 3, 3, activation='relu', padding='same')

    out = concatenate([conv3x3, conv5x5, conv7x7], axis=3)
    out = BatchNormalization(axis=3)(out)

    out = add([shortcut, out])
    out = Activation('relu')(out)
    out = BatchNormalization(axis=3)(out)

    return out

# ResPath
def ResPath(filters, length, inp):
    '''
    ResPath as defined in the MultiResUNet architecture, used for residual connections.
    Arguments:
        filters: Number of filters.
        length: Length of the ResPath.
        inp: Input keras layer.
    Returns:
        Output keras layer after applying ResPath.
    '''
    shortcut = inp
    shortcut = conv2d_bn(shortcut, filters, 1, 1, activation=None, padding='same')

    out = conv2d_bn(inp, filters, 3, 3, activation='relu', padding='same')
    out = add([shortcut, out])
    out = Activation('relu')(out)
    out = BatchNormalization(axis=3)(out)

    for i in range(length - 1):
        shortcut = out
        shortcut = conv2d_bn(shortcut, filters, 1, 1, activation=None, padding='same')
        out = conv2d_bn(out, filters, 3, 3, activation='relu', padding='same')
        out = add([shortcut, out])
        out = Activation('relu')(out)
        out = BatchNormalization(axis=3)(out)

    return out

# MultiResUnet model definition
def MultiResUnet(height, width, n_channels):
    '''
    Define the MultiResUNet model.
    Arguments:
        height: Height of the input image.
        width: Width of the input image.
        n_channels: Number of channels in the input image.
    Returns:
        Constructed MultiResUNet keras model.
    '''
    inputs = Input((height, width, n_channels))

    # MultiRes blocks and pooling layers
```

```
    mresblock1 = MultiResBlock(32, inputs)
    pool1 = MaxPooling2D(pool_size=(2, 2))(mresblock1)
    mresblock1 = ResPath(32, 4, mresblock1)

    mresblock2 = MultiResBlock(32*2, pool1)
    pool2 = MaxPooling2D(pool_size=(2, 2))(mresblock2)
    mresblock2 = ResPath(32*2, 3, mresblock2)

    mresblock3 = MultiResBlock(32*4, pool2)
    pool3 = MaxPooling2D(pool_size=(2, 2))(mresblock3)
    mresblock3 = ResPath(32*4, 2, mresblock3)

    mresblock4 = MultiResBlock(32*8, pool3)
    pool4 = MaxPooling2D(pool_size=(2, 2))(mresblock4)
    mresblock4 = ResPath(32*8, 1, mresblock4)

    mresblock5 = MultiResBlock(32*16, pool4)

    # Upsampling and concatenation
    up6 = concatenate([Conv2DTranspose(32*8, (2, 2), strides=(2, 2), padding='same')(
    mresblock5), mresblock4], axis=3)
    mresblock6 = MultiResBlock(32*8, up6)

    up7 = concatenate([Conv2DTranspose(32*4, (2, 2), strides=(2, 2), padding='same')(
    mresblock6), mresblock3], axis=3)
    mresblock7 = MultiResBlock(32*4, up7)

    up8 = concatenate([Conv2DTranspose(32*2, (2, 2), strides=(2, 2), padding='same')(
    mresblock7), mresblock2], axis=3)
    mresblock8 = MultiResBlock(32*2, up8)

    up9 = concatenate([Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='same')(
    mresblock8), mresblock1], axis=3)
    mresblock9 = MultiResBlock(32, up9)

    # Final convolutional layer
    conv10 = conv2d_bn(mresblock9, 1, 1, 1, activation='sigmoid')

    # Model compilation
    model = Model(inputs=[inputs], outputs=[conv10])
    return model

# Custom metrics for model evaluation
def dice_coef(y_true, y_pred):
    '''
    Calculate the Dice Coefficient for model evaluation.
    Arguments:
        y_true: True labels.
        y_pred: Predicted labels.
    Returns:
        Dice Coefficient as a float.
    '''
    smooth = 1.0
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = K.sum(y_true_f * y_pred_f)
    return (2. * intersection + smooth) / (K.sum(y_true_f) + K.sum(y_pred_f) + smooth)

def jacard(y_true, y_pred):
    '''
    Calculate the Jaccard Index (IoU) for model evaluation.
    Arguments:
        y_true: True labels.
        y_pred: Predicted labels.
    Returns:
        Jaccard Index as a float.
    '''
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = K.sum(y_true_f * y_pred_f)
    union = K.sum(y_true_f + y_pred_f - y_true_f * y_pred_f)
```

```
    return intersection / union

# Model saving function
def saveModel(model, training_name):
    '''
    Save the model architecture and weights.
    Arguments:
        model: The trained keras model.
        training_name: Name of the training session for directory structuring.
    '''
    model_json = model.to_json()
    try:
        os.makedirs(f'/home/thijsbolscher/Documents/Aerospace/Thesis/Testing/MultiResUNet-Rocks/checkpoints/{training_name}/models')
    except FileExistsError:
        pass

    with open(f'/home/thijsbolscher/Documents/Aerospace/Thesis/Testing/MultiResUNet-Rocks/checkpoints/{training_name}/models/modelP.json', 'w') as fp:
        fp.write(model_json)

    model.save_weights(f'/home/thijsbolscher/Documents/Aerospace/Thesis/Testing/MultiResUNet-Rocks/checkpoints/{training_name}/models/modelW.h5')
    model.save(f'/home/thijsbolscher/Documents/Aerospace/Thesis/Testing/MultiResUNet-Rocks/checkpoints/{training_name}/models/complete_model.h5')
    print('Model saved in "models" folder.')



def evaluateModel(model, X_test, Y_test, batchSize, training_name, original_images_test):
    # Directory creation for saving resulting images and model data
    os.makedirs(f'/home/thijsbolscher/Documents/Aerospace/Thesis/Testing/MultiResUNet-Rocks/checkpoints/{training_name}/resulting_imgs', exist_ok=True)
    os.makedirs(f'/home/thijsbolscher/Documents/Aerospace/Thesis/Testing/MultiResUNet-Rocks/checkpoints/{training_name}/models', exist_ok=True)

    fig_path = f'/home/thijsbolscher/Documents/Aerospace/Thesis/Testing/MultiResUNet-Rocks/checkpoints/{training_name}/resulting_imgs/'

    # Model prediction and evaluation
    yp = model.predict(x=X_test, batch_size=batchSize, verbose=1)
    val_loss = model.evaluate(X_test, Y_test, batch_size=batchSize, verbose=0)
    yp = np.round(yp, 0)

    # Visualization of predictions
    for i in range(10):
        plt.figure(figsize=(20, 10))
        plt.subplot(1, 3, 1)
        plt.imshow(original_images_test[i])  # Input image
        plt.title('Input')
        plt.subplot(1, 3, 2)
        plt.imshow(Y_test[i].reshape(Y_test[i].shape[0], Y_test[i].shape[1]).astype(np.int8))  # Ground truth
        plt.title('Ground Truth')
        plt.subplot(1, 3, 3)
        plt.imshow(yp[i].reshape(yp[i].shape[0], yp[i].shape[1]).astype(np.int8))  # Prediction
        plt.title('Prediction')

        intersection = yp[i].ravel() * Y_test[i].ravel()
        union = yp[i].ravel() + Y_test[i].ravel() - intersection
        jacard_index = np.sum(intersection) / np.sum(union)
        plt.suptitle(f'Jaccard Index: {np.sum(intersection)}/{np.sum(union)} = {jacard_index}')
        plt.savefig(f'{fig_path}{i}.png', format='png')
        plt.close()

    # Metric calculation for the entire test dataset
    jacard, dice = 0, 0
    recall_scores, precision_scores, f1_scores, accuracy_scores = [], [], [], []

```

```
    for i in range(len(Y_test)):
        yp_flat = yp[i].ravel()
        y_true_flat = Y_test[i].ravel()
        intersection = yp_flat * y_true_flat
        union = yp_flat + y_true_flat - intersection

        jacard += np.sum(intersection) / np.sum(union)
        dice += 2. * np.sum(intersection) / (np.sum(yp_flat) + np.sum(y_true_flat))

        tp, fp, fn = np.sum(intersection), np.sum(yp_flat) - np.sum(intersection), np.sum(y_true_flat) - np.sum(intersection)
        tn = len(y_true_flat) - (tp + fp + fn)

        accuracy = (tp + tn) / (tp + tn + fp + fn)
        recall = tp / (tp + fn) if (tp + fn) != 0 else 0
        precision = tp / (tp + fp) if (tp + fp) != 0 else 0
        f1 = 2 * (precision * recall) / (precision + recall) if (precision + recall) != 0 else 0

        accuracy_scores.append(accuracy)
        recall_scores.append(recall)
        precision_scores.append(precision)
        f1_scores.append(f1)

    # Averaging the metrics
    avg_accuracy = np.mean(accuracy_scores)
    avg_recall = np.mean(recall_scores)
    avg_precision = np.mean(precision_scores)
    avg_f1 = np.mean(f1_scores)

    # Printing the average metrics
    print(f"Average Accuracy: {avg_accuracy:.4f}, Average Recall: {avg_recall:.4f}, Average Precision: {avg_precision:.4f},  Average F1 Score: {avg_f1:.4f}")

    ######################################

    jacard /= len(Y_test)
    dice /= len(Y_test)
    #precision = precision_m()
    #recall /= len(Y_test)
    #f1 /= len(Y_test)


    print('Jacard Index : '+str(jacard))
    print('Dice Coefficient : '+str(dice))


    fp = open('/home/thijsbolscher/Documents/Aerospace/Thesis/Testing/MultiResUNet-Rocks/checkpoints/' +training_name +'/models/log.txt','a')
    fp.write(str(jacard)+'\n')
    fp.close()

    fp = open('/home/thijsbolscher/Documents/Aerospace/Thesis/Testing/MultiResUNet-Rocks/checkpoints/' +training_name + '/models/best.txt','r')
    best = fp.read()
    fp.close()

    if(jacard>float(best)):
        print('***********************************************')
        print('Jacard Index improved from '+str(best)+' to '+str(jacard))
        print('***********************************************')
        fp = open('/home/thijsbolscher/Documents/Aerospace/Thesis/Testing/MultiResUNet-Rocks/checkpoints/' +training_name + '/models/best.txt','w')
        fp.write(str(jacard))
        fp.close()

        saveModel(model)

    return(dice, jacard, avg_accuracy, avg_precision, avg_recall, avg_f1, val_loss)


```

```
363
364 def trainStep(model, X_train, Y_train, X_test, Y_test, epochs, batchSize, training_name):
365     '''
366     Function to train a machine learning model and evaluate it at each epoch.
367
368     Parameters:
369         model (keras.Model): The machine learning model to be trained.
370         X_train, Y_train: Training data and labels.
371         X_test, Y_test: Validation data and labels.
372         epochs (int): Number of epochs to train the model.
373         batchSize (int): Size of the batch used in training.
374         training_name (str): Name identifier for the training session, used for logging and
      checkpointing.
375     '''
376
377     # Setting up TensorBoard logging and checkpoints
378     log_dir = os.path.join("tensorboard_logs", training_name, "cp.ckpt" + datetime.datetime
    .now().strftime("%Y%m%d-%H%M%S"))
379     tensorboard_callback = TensorBoard(log_dir=log_dir, histogram_freq=1)
380
381     checkpoint_path = os.path.join("checkpoints", training_name, "cp.ckpt")
382     cp_callback = ModelCheckpoint(filepath=checkpoint_path, save_best_only=False, verbose
    =1, save_weights_only=False)
383
384     # Lists to store metrics
385     loss_history, val_loss_history, dice_coef_history, jacard_history = [], [], [], []
386     accuracy_history, precision_history, recall_history, f1_history = [], [], [], []
387
388     # Training loop
389     for epoch in range(epochs):
390         gc.collect()
391         print('Now we start training:', 'Epoch {}'.format(epoch + 1))
392         history = model.fit(X_train, Y_train, batch_size=batchSize, epochs=1, verbose=1,
    callbacks=[tensorboard_callback, cp_callback])
393
394         # Recording training loss
395         loss_history.append(history.history['loss'][0])
396         print('Evaluating model performance:', 'Epoch {}'.format(epoch + 1))
397
398         # Evaluating the model on validation data
399         dice_coef, jacard, accuracy, precision, recall, f1, val_loss = evaluateModel(model,
      X_test, Y_test, batchSize, training_name)
400
401         # Recording validation metrics
402         dice_coef_history.append(dice_coef)
403         jacard_history.append(jacard)
404         accuracy_history.append(accuracy)
405         precision_history.append(precision)
406         recall_history.append(recall)
407         f1_history.append(f1)
408         val_loss_history.append(val_loss[0])  # Assuming val_loss is a tuple/list
409
410         print('Validation loss history:', val_loss_history)
411
412         # Plotting training and validation loss
413         plt.figure(figsize=(10, 6))
414         plt.plot(range(1, len(loss_history) + 1), loss_history, label='Training Loss')
415         plt.plot(range(1, len(val_loss_history) + 1), val_loss_history, label='Validation
    Loss')
416         plt.title('Training and Validation Loss Over Epochs')
417         plt.xlabel('Epoch')
418         plt.ylabel('Loss')
419         plt.legend()
420         plt.grid(True)
421         plt.show()
422
423     return model
424
425
426 # Custom metric functions
427 def recall_m(y_true, y_pred):
```

```
    '''
    Calculate the recall metric.
    Arguments:
        y_true: True labels.
        y_pred: Predicted labels.
    Returns:
        Calculated recall value.
    '''
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

def precision_m(y_true, y_pred):
    '''
    Calculate the precision metric.
    Arguments:
        y_true: True labels.
        y_pred: Predicted labels.
    Returns:
        Calculated precision value.
    '''
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

def f1_m(y_true, y_pred):
    '''
    Calculate the F1 score.
    Arguments:
        y_true: True labels.
        y_pred: Predicted labels.
    Returns:
        Calculated F1 score.
    '''
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)
    return 2 * ((precision * recall) / (precision + recall + K.epsilon()))


##########################################################################################
# Training setup
training_name = 'Mars_FP16_BS4'
model = MultiResUnet(height=192, width=256, n_channels=3)

# Compile the model with custom metrics
model.compile(optimizer='adam', loss='binary_crossentropy',
              metrics=[dice_coef, jacard, 'accuracy', keras.metrics.Precision(), keras.
    metrics.Recall()])

# Save the initial model configuration
saveModel(model, training_name)

# Initialize log files
with open(f'/home/thijsbolscher/Documents/Aerospace/Thesis/Testing/MultiResUNet-Rocks/
    checkpoints/{training_name}/models/log.txt', 'w') as fp:
    pass
with open(f'/home/thijsbolscher/Documents/Aerospace/Thesis/Testing/MultiResUNet-Rocks/
    checkpoints/{training_name}/models/best.txt', 'w') as fp:
    fp.write('-1.0')

# Start the training process
trainStep(model, X_train, Y_train, X_test, Y_test, epochs=15, batchSize=4, training_name)

##########################################################################################
# Code to load and continue training the model for a new training round
new_training_name = 'FP16_first_round'
checkpoint_dir = os.path.join("/home/thijsbolscher/Documents/Aerospace/Thesis/Testing/
    MultiResUNet-Rocks/checkpoints", new_training_name, 'cp.ckpt')

```

```
# Load the model from the checkpoint
print(f"Loading model from {checkpoint_dir}")
model = tf.keras.models.load_model(checkpoint_dir, custom_objects={'dice_coef': dice_coef, 'jacard': jacard})

# Re-compile the model with the same settings
model.compile(optimizer='adam', loss='binary_crossentropy',
              metrics=[dice_coef, jacard, 'accuracy', keras.metrics.Precision(), keras.metrics.Recall()])

# Continue training
trainStep(model, X_train, Y_train, X_test, Y_test, epochs=200, batchSize=2, training_name=new_training_name)
```

Source Code C.2: Training MultiResUNet.

C.3. Transforming PNG Images to BIN for Inference on Myriad

```

'''
Description: This script is designed for processing and converting image data to a planar format in FP16. It reads images from a specified source directory, resizes them to a target size, and splits them into individual color channels (BGR). These channels are then interleaved and saved in a binary format to a destination directory. This process is crucial for preparing images for specific machine learning or image processing tasks where planar format and reduced precision (FP16) are beneficial.
Author: Thijs Bolscher (thijsbolscher18@live.nl)
Creation Date: August/September 2023
'''

import cv2
import numpy as np
import os
from pathlib import Path
from tqdm import tqdm

# Define source and destination directories
src_dir = Path("validation_images")
dst_dir = Path("/home/thijsbolscher/Documents/Aerospace/Thesis/Testing/Image_conversion/bin_images_planar_fp16")

# Retrieve image file names from source directories
img_files = sorted(next(os.walk(src_dir))[2])

# Initialize lists for storing image data
X = []  # List for resized images

# Define the target size for image resizing
target_size = (256, 192)

# Create the destination directory if it doesn't exist
os.makedirs(dst_dir, exist_ok=True)

# Process each image file
for img_fl in tqdm(img_files):
    if img_fl.endswith('.png'):
        # Define the source path
        src_path = src_dir / img_fl

        # Read and resize the image to the target size
        img = cv2.imread(str(src_path), cv2.IMREAD_COLOR)
        resized_img = cv2.resize(img, target_size, interpolation=cv2.INTER_CUBIC).astype(np.float16)

        # Append resized image to the list
        X.append(resized_img)

        # Split the image into separate color channels (BGR format)
        blue_channel, green_channel, red_channel = cv2.split(resized_img)
```

```

        try:
            # Interleave the color channels (planar format)
            planar_data = np.stack((blue_channel, green_channel, red_channel))

            # Transpose the data to rearrange channel order
            planar_data = planar_data.transpose((1, 2, 0))

            # Set the destination path with '.bin' extension
            dst_path = dst_dir / src_path.with_suffix(".bin").name

            # Save the interleaved FP16 pixel data to a binary file
            planar_data.tofile(dst_path)
        except Exception as e:
            print(f"Error converting {src_path}: {str(e)}")

# Print the shape of the last processed image's planar data
print(f"Shape of the last processed image's planar data: {np.shape(planar_data)}")
```

Source Code C.3: Training MultiResUNet.

C.4. Transforming OMX output files to JPEG after Inference on Myriad

```

    import os
import glob
import numpy as np
import cv2

# Paths for the source tensors and destination for reconstructed images
tensor_folder_path = '/home/thijsbolscher/CogniSatApp/CogniSatApp/apps/DM_MultiResUNet/outputTensors'
output_folder_path = '/home/thijsbolscher/CogniSatApp/CogniSatApp/apps/DM_MultiResUNet/reconstructed_images'

# Configure NumPy to display the entire array when printing
np.set_printoptions(threshold=np.inf)

# Get a list of .mxo files in the specified directory
mxo_files = glob.glob(os.path.join(tensor_folder_path, '*.mxo'))

# Threshold value for image conversion
threshold_value = 1

# Iterate over each .mxo file
for tensor_file in mxo_files:
    print(f"Processing {tensor_file}")

    # Read mask data from the .mxo file
    mxo_data = np.fromfile(tensor_file, dtype=np.float16)

    # Scale the data to the range [0, 255] and convert to uint8
    mxo_data = (mxo_data * 255).astype(np.uint8)

    # Optional thresholding (currently commented out)
    # mxo_data = ((mxo_data >= threshold_value).astype(np.uint8) * 255)

    # Reshape the data to the desired image dimensions (here, 512x512)
    mask = mxo_data.reshape((512, 512))

    # Construct the output file path with .jpeg extension
    output_file = os.path.join(output_folder_path, os.path.splitext(os.path.basename(tensor_file))[0] + '.jpeg')

    # Save the image using OpenCV
    cv2.imwrite(output_file, mask)
```

```
    print(f"Saved {output_file}")
```

Source Code C.4: Transforming OMX output files to JPEG after Inference on Myriad.

C.5. The Endurance Test for Energy and Power Consumption

```
'''
Description: This script models the power consumption and energy usage of an On-Board
    Computer (OBC) in a space exploration context. It simulates the power usage over time
    considering various operational tasks such as image preprocessing, object detection,
    and path planning. The script also visualizes the OBC's energy consumption and
    remaining energy over time, which is crucial for power management in space missions.
Author: Thijs Bolscher (thijsbolscher18@live.nl)
Creation Date: August/September 2023
'''

import numpy as np
import matplotlib.pyplot as plt
import time

# Depth of Discharge and Energy Calculation
DoD = 80/100
energy_available_per_hour = 0.1 * 5 * DoD * 0.0475 * 243  # Wh
energy_available_per_hour_joule = energy_available_per_hour * 3600  # Convert to Joules



# Function to simulate and visualize energy usage
def Energy_at_t_graph_milliseconds(simulation_time, XE2_onboard, cycle_time,
    energy_available):
    energies = []
    power_per_second = []

    # Power consumption data (in Joules per millisecond)
    nominal_power = 2e-3
    nominal_power_incl_screen = 1.54e-3
    XE2_idle = 4.04e-3 - nominal_power_incl_screen
    OBC_tasks_power = 0.5e-3
    PP_power = 3.06e-3 - nominal_power_incl_screen
    preprocessing_power = 2.15e-3 - nominal_power_incl_screen
    Loading_package_power = 1.95e-3 - nominal_power_incl_screen

    # Time data (in milliseconds)
    startuptime_XE2 = 6e3
    PP_duration = 1e3
    Preprocessing_time = 0.15e3 * 2
    Loading_package_time = 7.1e3 + 2e3
    firmware_time_XE2 = 12.92e3

    ###########################################################################
    #Dependent on board
    if XE2_onboard == False:
        OD_duration = 56.22e3 * N_rovers *2
        OD_power = (3.06e-3 - nominal_power_incl_screen )   # W    sizing!
        FDIR_duration = (56.22e3 * N_rovers)/10 *2 #s


    else:
        OD_duration = 0.34e3 * 6.25 *  N_rovers *2  #s #######Change this line if the rover
     needs to run inference on more individuals
        OD_power = 4.84e-3 - nominal_power_incl_screen #W
        FDIR_duration =  (0.34e3 * 6.25 *  N_rovers)/10 *2  #ms
        #for the XE2 this is loading firmware and network to XE2, not required in warm boot
        Loading_firmware_power = 4.03e-3 -nominal_power_incl_screen #J/ms

    ###########################################################################
    energy_left = energy_available
    # Simulation loop for energy usage over time
    for t in range(simulation_time):
```

```
        power_use_this_second = 0

        # Subtract nominal power and add to power usage
        energy_left -= nominal_power
        power_use_this_second += nominal_power

#         #Load packages RP2 ######################
        if t % cycle_time < (Loading_package_time):
            energy_left -= Loading_package_power
            power_use_this_second += Loading_package_power

        #preprocessing ####################3
        if t % cycle_time > (Loading_package_time) and t % cycle_time < (Preprocessing_time + Loading_package_time): #
            energy_left -= preprocessing_power
            power_use_this_second += preprocessing_power

        #Startup XE2 ###################################33
        if t % cycle_time >(Preprocessing_time + Loading_package_time  -(startuptime_XE2)) and t%cycle_time <(Preprocessing_time +Loading_package_time ) and XE2_onboard == True: #include startuptime of XE2
            energy_left -=  XE2_idle
            power_use_this_second += XE2_idle

        #Load Firmware and model XE2 ##########################
        if t % cycle_time >(Preprocessing_time + Loading_package_time  ) and t%cycle_time <(Preprocessing_time +Loading_package_time + firmware_time_XE2) and XE2_onboard == True : #include startuptime of XE2
            energy_left -=  Loading_firmware_power
            power_use_this_second += Loading_firmware_power


        #Rock Detection ########################################
        if XE2_onboard == True and t % cycle_time > (Preprocessing_time + Loading_package_time + firmware_time_XE2) and t % cycle_time < (OD_duration+ Preprocessing_time +Loading_package_time + firmware_time_XE2): #OD on 2 images every cycle_time seconds after preprocessing with one second break
            energy_left -=  OD_power
            power_use_this_second += OD_power

        elif XE2_onboard ==False and t % cycle_time > (Preprocessing_time + Loading_package_time ) and t % cycle_time < (OD_duration+ Preprocessing_time + Loading_package_time ) :
            energy_left -=  OD_power
            power_use_this_second += OD_power

        #FDIR ##############################################
        if XE2_onboard == True and t % cycle_time > (Preprocessing_time + Loading_package_time + OD_duration +firmware_time_XE2  ) and t % cycle_time < ( PP_duration + Preprocessing_time + Loading_package_time + OD_duration + firmware_time_XE2 + FDIR_duration):
            energy_left -=  OD_power
            power_use_this_second += OD_power

        elif XE2_onboard ==False and t % cycle_time > (Preprocessing_time + Loading_package_time + OD_duration   ) and t % cycle_time < (PP_duration + Preprocessing_time + Loading_package_time + OD_duration  + FDIR_duration):
            energy_left -=  OD_power
            power_use_this_second += OD_power

        #Path Planning Correct
        if XE2_onboard == True and t % cycle_time > (Preprocessing_time + Loading_package_time + OD_duration +firmware_time_XE2 +FDIR_duration +1e3 ) and t % cycle_time < (PP_duration + Preprocessing_time + Loading_package_time + OD_duration + firmware_time_XE2+FDIR_duration + 1e3):
            energy_left -=  PP_power
            power_use_this_second += PP_power

        elif XE2_onboard ==False and t % cycle_time > (Preprocessing_time + Loading_package_time + OD_duration + FDIR_duration  + 1e3 ) and t % cycle_time < ( PP_duration + Preprocessing_time + Loading_package_time + OD_duration +FDIR_duration +
```

```
    1e3):
109             energy_left -=  PP_power
110             power_use_this_second += PP_power
111 
112 
113 
114         energies.append(energy_left)
115         power_per_second.append(power_use_this_second)
116 
117     # Printing peak and nominal power usage
118     print("Peak power during operational cycle =", 1000 * max(power_per_second), "mW")
119     print("Nominal power during operational cycle =", 1000 * min(power_per_second), "mW")
120 
121     # Plotting the results
122     adjusted_time = [time_val / 1000 for time_val in range(simulation_time)]
123     adjusted_power = [power_val * 1000 for power_val in power_per_second]
124 
125     plt.figure(figsize=(12, 5))
126     plt.subplot(1, 2, 1)
127     plt.plot(adjusted_time, adjusted_power, color='orange')
128     plt.xlabel('Time (seconds)')
129     plt.ylabel('Power (Watts)')
130     plt.title('OBC Power Consumption Over Time')
131     plt.grid(True)
132 
133     plt.subplot(1, 2, 2)
134     plt.plot(adjusted_time, energies)
135     plt.xlabel('Time (seconds)')
136     plt.ylabel('Energy (Joules)')
137     plt.title('The Energy Available for the OBC Over Time')
138     plt.grid(True)
139     plt.tight_layout()
140     plt.show()
141 
142     print('Energy left at the end of', t, 'seconds =', energy_left, 'Joules')
143     print('Energy consumed in the total interval =', energy_available_per_hour_joule -
    energy_left, 'Joules')
144     return energy_left
145 
146 
147 
148 
149 ##############################################################################################
150 ################### WARMBOOT/LOW-POWER MODE ##################################################
151 ##############################################################################################
152 
153 
154 def Energy_at_t_graph_milliseconds_warmboot(simulation_time, XE2_onboard, cycle_time,
    energy_available):
155     energies = []
156     power_per_second = []
157 
158 #     cycle_time = 240e3 #Check with Raj.
159     N_rovers = 1   #we change this if HETEROGENEOUS swarm
160 
161     #Power Data
162     nominal_power = 3e-3 #J/ms
163     nominal_power_incl_screen = 1.54e-3 #J/ms
164     XE2_idle = (4.04e-3 - nominal_power_incl_screen)  #W
165     OBC_tasks_power = 0.5e-3 #J/ms
166     PP_power = 3.06e-3 -nominal_power_incl_screen #J/ms
167     preprocessing_power = 2.15e-3 - nominal_power_incl_screen #J/ms
168     Loading_package_power = 1.95e-3 - nominal_power_incl_screen #J/ms
169 
170     #Time Data
171     startuptime_XE2 = 6e3 #s
172     PP_duration = 1e3 #millisecond ,  see requirements on page 66 of thomas' thesis we can
    ask Thomas Manteaux for broad estimation of path planning  from what he experienced on
    his chip
173     Preprocessing_time = 0.15e3 *N_rovers  #millisecond multiply by two for two images.
174     Loading_package_time = 7.1e3  #ms
```

```
    firmware_time_XE2 = 12.92e3  #ms

    ###########################################################################
    #Dependent on board
    if XE2_onboard == False:
        OD_duration = 56.22e3 * N_rovers
        OD_power = (3.06e-3 - nominal_power_incl_screen )   # W    sizing!
        FDIR_duration = OD_duration/10 *2*N_rovers #s


    else:
        OD_duration = 0.34e3 * 6.25 *  N_rovers  #s #######Change this line if the rover needs to run inference on more individuals
        OD_power = 4.84e-3 - nominal_power_incl_screen #W
        FDIR_duration = OD_duration/10 *2 * N_rovers #ms
        #for the XE2 this is loading firmware and network to XE2, not required in warm boot
        Loading_firmware_power = 4.03e-3 -nominal_power_incl_screen #J/ms

    ###########################################################################
    energy_left = energy_available
    for t in range(simulation_time):
        power_use_this_second = 0 #we will use this to calculate the power use at every second

        energy_left -= nominal_power
        power_use_this_second += (nominal_power)


        #Load packages RP2 ######################
        if t % cycle_time < (Loading_package_time):
            energy_left -= Loading_package_power
            power_use_this_second += Loading_package_power

        #preprocessing ####################3
        if t % cycle_time > (Loading_package_time) and t % cycle_time < (Preprocessing_time + Loading_package_time): #
            energy_left -= preprocessing_power
            power_use_this_second += preprocessing_power



        #Rock Detection ########################################
        if XE2_onboard == True and t % cycle_time > (Preprocessing_time + Loading_package_time ) and t % cycle_time < (OD_duration+ Preprocessing_time + Loading_package_time ): #OD on 2 images every cycle_time seconds after preprocessing with one second break
            energy_left -=  OD_power
            power_use_this_second += OD_power


        #FDIR ##############################################
        if XE2_onboard == True and t % cycle_time > (Preprocessing_time + Loading_package_time + OD_duration   ) and t % cycle_time < (PP_duration + Preprocessing_time + Loading_package_time + OD_duration  + FDIR_duration):
            energy_left -=  OD_power
            power_use_this_second += OD_power



        #Path Planning Correct
        if XE2_onboard == True and t % cycle_time > (Preprocessing_time + Loading_package_time + OD_duration +FDIR_duration +1e3 ) and t % cycle_time < ( PP_duration + Preprocessing_time + Loading_package_time + OD_duration +FDIR_duration + 1e3):
            energy_left -=  PP_power
            power_use_this_second += PP_power
```

```
235         energies.append(energy_left)
236         power_per_second.append(power_use_this_second)
237
238
239     print("Peak power during operational cycle =", max(power_per_second))
240     print("Nominal power during operational cycle =", min(power_per_second))
241
242     # Adjusting data for plotting
243     adjusted_time = [time_val / 1000 for time_val in range(simulation_time)]
244     adjusted_power = [power_val * 1000 for power_val in power_per_second]
245
246     # Create two subplots side by side
247     plt.figure(figsize=(12, 5))  # Adjust the figure size as needed
248
249     # Plotting power graph on the right
250     plt.subplot(1, 2, 1)
251     plt.plot(adjusted_time, adjusted_power, color='orange')  # Plotting adjusted power
        usage
252     plt.xlabel('Time (seconds)')
253     plt.ylabel('Power (Watts)')
254     plt.ylim(2, None)  # Set the minimum y-value to 2 and let maximum be determined
        automatically
255     plt.title('Power Usage Over Time')
256     plt.grid(True)
257
258     # Plotting energy graph on the left
259     plt.subplot(1, 2, 2)
260     plt.plot(adjusted_time, energies)
261     plt.xlabel('Time (seconds)')
262     plt.ylabel('Energy (Joules)')
263     plt.title('The Energy Available for the OBC Over Time')
264     plt.grid(True)
265
266
267     plt.tight_layout()
268     plt.show()
269
270     plt.plot(adjusted_time, adjusted_power, color='orange')  # Plotting adjusted power
        usage
271     plt.xlabel('Time (seconds)')
272     plt.ylabel('Power (Watts)')
273     plt.ylim(2, None)  # Set the minimum y-value to 2 and let maximum be determined
        automatically
274     plt.title('Power Consumption during one Operational Cycle')
275     plt.show()
276
277     print('Energy left at the end of', t, 'seconds = ', energy_left, 'Joules')
278     print('Energy consumed in the total interval =', energy_available_per_hour_joule-
        energy_left)
279     return energy_left
```

Source Code C.5: The Endurance Test for Energy and Power Consumption.