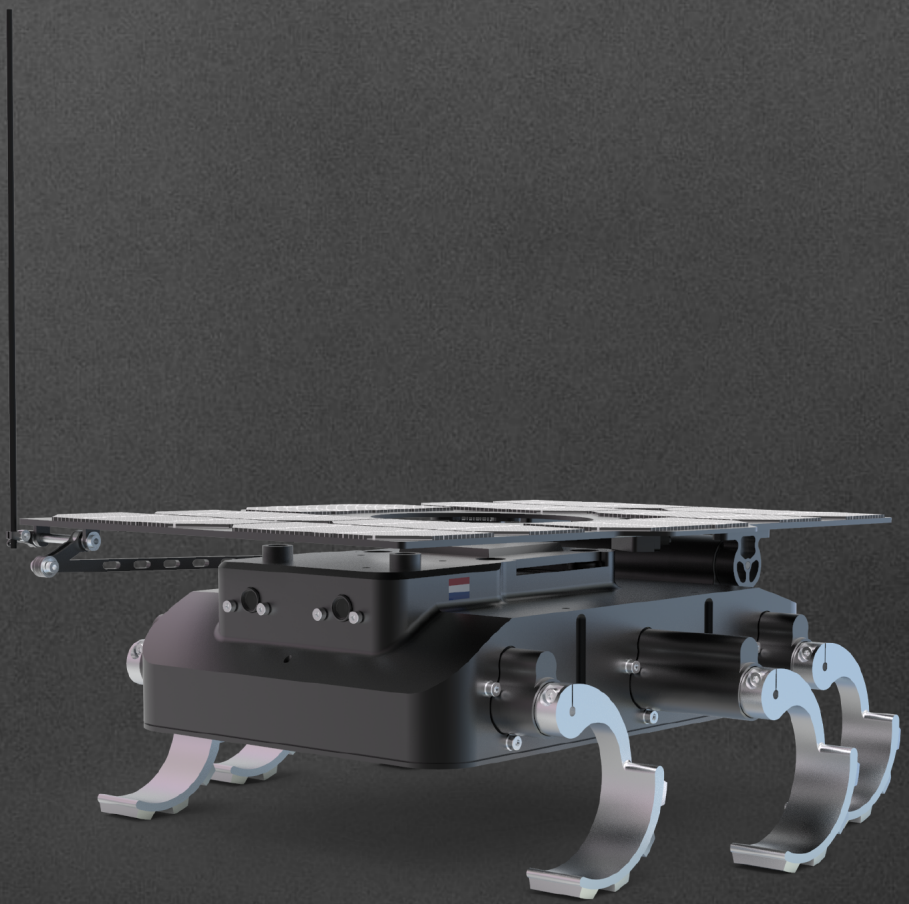
OPAL

A Stereo Vision Obstacle Processing ALgorithm for a Walking Lunar Rover

S.P. Rovers

Technische Universiteit Delft



TU Delft

LUNAR ZEBRO + subsultio aude ad

OPAL

A Stereo Vision Obstacle Processing ALgorithm for a Walking Lunar Rover

by

S.P. Rovers

to obtain the degree of Master of Science
at the Delft University of Technology,

Student number: 4206193
Project duration: October 16, 2019 – June 30, 2021
Thesis committee:

Dr. S. Speretta,	TU Delft, Daily Supervisor
Dr. A. Cervone,	TU Delft, Chair
ir. C. de Wagter,	TU Delft, Examiner
ir. Dr. C.J.M. Verhoeven,	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Summary

The Lunar Zebro is a small six-legged robot. It has the potential to be used in a swarm carrying out objectives like exploring planetary surfaces. A new step towards autonomous navigation is made with the newly developed Obstacle Processing ALgorithm (OPAL) using primarily open-source libraries. This study showed that the initial iteration of OPAL could detect rocks and determine their absolute position to the rover's low-positioned cameras using a stereo vision system. Obstacles and their relative distances are detected using the disparity map—the amount of shift of pixels in the stereo image pair. When translating the disparity to V-disparity, a histogram of the disparity per row, the ground and the obstacle could be isolated. It took six steps to realise this thesis goal. After setting the requirements, a test model, called Bars, was developed and tested at a location containing a Mars-like environment (Decos). This test model uses, along with Lunar Zebro's hardware, mostly Commercial Off-The-Shelf products. With the footage, all the different components of OPAL were integrated into one algorithm. Hereafter, a pipeline on a server was created, and multiple test cases were run to establish results. The predetermined requirements of the algorithm were validated using measuring tape measurements and ground-truth bounding boxes tracked by a CSRT-algorithm. Together, Bars and the initial iteration of OPAL prove feasibility and expose opportunities and challenges, which could be a starting point for optimisations or other approaches.

Keywords: OPAL, Bars, Stereo Vision, Obstacle Detection, Lunar Zebro

Preface

This thesis is part of the Space Engineering Profile Program within the Space Flight Track of the Master Aerospace Engineering at the Delft University of Technology. In the past 20 months, much work has been done to pass this 42 ECTS course. Within this crazy period, the world turned upside down due to Covid-19. Luckily this pandemic seems under control, and the world can get up to speed again. My thesis period will end with this thesis and the upcoming thesis defence. I would like to thank my family and friends for their support. This big thank you is also for my daily supervisor Stefano, who helped and supported me during this thesis. Thanks!

S.P. Rovers
Delft, June 2021

Contents

List of Figures

List of Tables

List of Symbols

The following list describes several symbols that will be used within this document

ΔP_d	Disparity error of point P
Δx	Pixel width
λ	A constant in the Disparity Energy Function
$\mathbf{F}$	Fundamental matrix
$\mathbf{H}$	Homograpy of a projected image
$\mathbf{u}$	Image coordinates
b	Base of the stereo setup
$E(d)$	Disparity Energy Function
f	Focal length and random signal
g	random signal
h	random signal
i	Index
j	Index
k	Index and number of iterations
p	Chance of finding model
P_z	Z-Coordinate of point P
w^n	Probability points are inliers
$x_L - x_r$	Disparity

List of Acronyms

The following list describes several acronyms that will be used within this document

AP	Average Precision
BMS	Battery Management System
CCD	Charge-Coupled Device
CI/CD	Continuous Integration, Continuous Delivery and Continuous Deployment
CLAHE	Contrast Limited Adaptive Histogram Equalization
CMOS	Complementary Metal Oxide Semiconductor
COTS	Commercial Off-The-Shelf
CSI	Camera Serial Interface
CSRT	Channel and Spatial Reliability Tracking
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DfA	Design for Assembly
EMM	EuroMoonMars
EPS	Electrical Power Subsystem
ESA	European Space Agency
ESD	Electro Static Discharge
FN	False Negative
FOV	Field of View
FP	False Positive
FRAM	Ferrolectric Random-Access Memory
FWMAV	Flapping-Wing Micro Air Vehicles
GESTALT	Grid-based Estimation of Surface Traversability Applied to Local Terrain
GUI	Graphical User Interface
I2C	Inter-Integrated Circuit
IMP	Imager for Mars Pathfinder
IMU	Inertial Measurement Unit
IOU	Intersection Over Union
LOP-G	Lunar Orbital Platform-Gateway
LUFAR	LUnar low Frequency ARray
MER	Mars Exploration Rover
NASA	National Aeronautics and Space Administration
NCC	Normalised Cross-Correlation
NRHO	Near-Rectilinear Halo Orbit
OBC	On-Board Computer
OPAL	Obstacle Processing ALgorithm
OpenCV	Open Source Computer Vision

P	Precision
R	Recall
RAM	Random-Access Memory
RANSAC	RANdom SAmple Consensus
RGB	Red, Green and Blue
RMSE	Root Mean Squared Error
ROI	Region Of Interest
RTK	Real-Time Kinematic positioning
SAD	Sum of Absolute Differences
SGBM	Semi-Global Block Matching
SGM	Semi-Global Matching
SHRIMP	Small High-Resolution Independent Modular Photographer
SIFT	Scale-Invariant Feature Transform
SSD	Sum of Squared Differences
TN	True Negative
TP	True Positive
UWB	Ultra-WideBand
Zebro	ZEsBenige RObot
ZPU	Zebro Processing Unit

1 Introduction

This chapter will provide the reader with background and a rationale for the Lunar Zebro project. It will form the basis of this thesis about the design and integration of a stereo vision obstacle detection algorithm named OPAL, which is an acronym for Obstacle Processing ALgorithm and its test system called Bars. At first, the Moon exploration market and the need for the Lunar Zebro is explored. Following this, the need and the research goals are stated. Finally, the research questions are elaborated on. At last, an introduction to the report is provided.

1.1. The Moon Exploration Market and the Need for the Lunar Zebro

After a 40 years break, the landing of China's Chang'e 3 mission of China has opened a new chapter in the history of exploring the Moon. More and more countries and companies are announcing missions to the surface of the Moon. Even the government of the United States is pushing the National Aeronautics and Space Administration (NASA) to get astronauts back on the Moon by 2024 [13]. The reason motivating these countries and companies is explained in the following section.

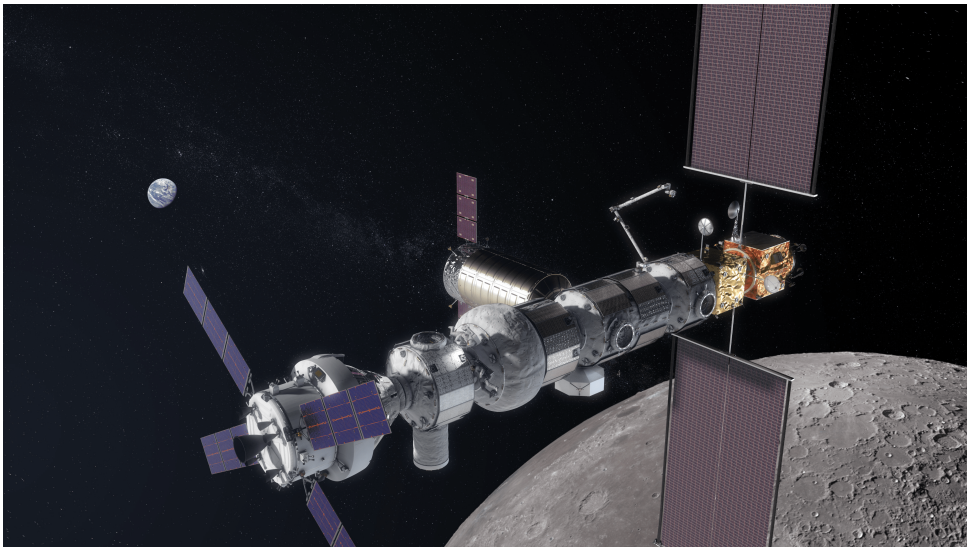


Figure 1.1: The Lunar Orbital Platform-Gateway (LOP-G) [36]

Recent research reveals that there is a presence of ice deposits in the shadowed parts of the craters on the south pole of the Moon [99]. This frozen water is attractive because it can be split into two elements: hydrogen and oxygen. Oxygen could help to refill the oxygen tanks of the astronauts, and hydrogen is a possible propellant for rockets. Therefore, this presence of ice makes the Moon an attractive gateway for exploring the universe. NASA, together with many other space agencies, are aiming to realise such a gateway known as the

Lunar Orbital Platform-Gateway (LOP-G). This LOP-G is shown in Figure 1.1. The LOP-G will be located in a Near-Rectilinear Halo Orbit (NRHO). This orbit is located around the Lagrangian point L1 of the Earth-Moon gravitational system and has as benefit that it requires not much travel energy to get to the surface of the Moon as well as escaping the Earth-Moon gravitational system [36]. This gateway will make research on the Moon and beyond much easier.

Furthermore, the Moon contains other interesting resources. Commodities such as helium-3 are stashed as molecules within the lunar soil. The molecules separate out when the soil is heated to high temperatures (above 600 degrees C) [100]. At the moment of writing this thesis, this harvest could be sold currently for around €1250 per gram. The reason for this is that helium-3 is a potential fuel for nuclear fusion. The output of nuclear fusion with helium-3 is likely to be less dangerous since it produces no radioactive waste, and it will have almost no carbon footprint [97]. Economically it is still not profitable considering the current cost of space travel. However, with the increasing amount of space travel and partners such as SpaceX, who are excelling in the development of reusable vehicles, costs for space travel will most likely drop in the near future.

The Moon holds additional scientific interest for investors. The soil and ice which are harvested for refuelling are coded with history of the universe. The temperature in the dark areas of craters on the poles of the Moon has never been warmer than 33 Kelvin [41]. These low temperatures could have preserved this matter from erosion or other degrading effects. Another promising scientific research area is the radio silent space on the far side of the Moon. The lack of atmosphere, absence of strong moon-quakes, and the radio silence make the Moon attractive for deep-space radio observations which cannot be conducted near Earth.

To conclude, there are much different science cases and other interest drivers, however, the costs for missions to the lunar surface are high. An opportunity for robots to carry out more work during a mission is by making use of swarming. Swarming rovers are small rovers that cooperate. This interaction of rovers could lead to interesting and useful swarm behaviour. To reduce cost and complexity, a swarm consists mostly of small robots. In planetary exploration, a small robot would be called a micro rover. Micro rovers have a required size of around 3000 x 2000 x 2000mm [62]. The mass of a micro rover is around four kilograms. Compared to other commonly used planetary rovers, these micro rovers are less expensive. Losing one does not have much impact, and the functionality will be easily taken over by the swarm. Hence, more risk could be taken during exploration of rough terrain or areas with extreme characteristics. This makes the swarm very reliable. This reliable multi-rover platform is also interesting, for example, form a scientific perspective because of its distributed sensor networks. A swarm can easily distribute itself into a fixed or dynamic network. One application, for example, is LUnar low Frequency ARray (LUFAR) where each rover is bearing an antenna to form an interferometric array of radio telescopes. This array, when deployed on the far side of the Moon, could measure cosmic noise. Scientists try to link these measurements to the history of the universe [5].

For these reasons, there is the need for micro rovers to be developed. This trend is also seen on the market, with the announcement of the development of micro rovers, as presented in Appendix A. The TU Delft saw the opportunity to take advantage of this new trend and redesigned one of their exploration and swarming robots — the Zebro. The ZEsBenige RObot (Zebro), literally translated six-legged robot, is the TU Delft adaptation of the bio-inspired six-legged rough-terrain robot RHex, that is developed by Boston Dynamics and Pen State University [9]. The redesigned Zebro is called the Lunar Zebro and is being developed at the faculty of Electrical Engineering, Mathematics and Computer Science at the Delft University of Technology. The goal of the Lunar Zebro is to provide hands on experience for students on lunar missions.

1.2. Thesis Need and Research Goals

Initially, the Lunar Zebro team wants to send one rover to the surface of the Moon. This rover will be a stepping-stone for swarming on the lunar surface and a chance to validate critical systems. For a successful mission, the Zebro will have to survive Moon conditions, transmit data back to Earth, and demonstrate walking. To walk on the Moon while surviving Moon conditions the rover will need to remain at operating temperature, and it needs to stay safe, maintain power, and connected while operating. A stereo vision obstacle detection system needs to be designed and tested in order to let the rover walk, while maintaining all the surviving requirements.

Nowadays, Most of the rover are using a mast to keep overview for navigation and obstacle avoidance, where

this new system needs to rely solely on one stereo vision camera system, positioned on the front of the body. This pilot study is to show that it is possible to develop such a detection system. The research objective of this thesis project is therefore summarised by the following sentence:

> The research objective of this study is to demonstrate an obstacle detection system for the Lunar Zebro that will be able to detect rocks and their absolute distance from the Zebro in the lunar environment by designing and building a test model, testing it on an analogue surface, designing and implementing an algorithm consisting of existing algorithms and techniques, and to verify whether this implementation meets the requirements to fulfil its purpose.

1.2.1. Research Questions

The thesis objective was further condensed into three research questions to direct this academic thesis. These questions are stated and elaborated in this section.

> **R1 -** How could obstacles and their absolute distance to the rover be detected, and how accurately does this distance need to be?

The first question is targeting the possibility of developing an obstacle detection system. It needs to be taken into account that the Lunar Zebro is designed to use a stereo vision system because a stereo vision system requires the lowest mass and power of all ranging sensors.

> **R2 -** How could the obstacle detection system be implemented to run on the Lunar Zebro and detect obstacles and their absolute distance to the rover from a low perspective, and how accurate can it determine distance?

The second question is about stitching multiple algorithm blocks together to create a result. The low perspective part and the limited hardware of the Lunar Zebro make the added scientific value since it makes detecting rock challenging. Also needs to be calculated what can actually which accuracy can be achieved. The third question below concerns the validation of the system. One of the problems here is the lack of test possibilities due to the Covid-19 pandemic.

> **R3 -** How to validate a low positioned stereo vision obstacle detection system in a lunar-like environment for a planetary rover in the Netherlands ?

1.3. Introduction to the Report

The remainder of this thesis report is divided into eight chapters. At first, in Chapter 2, some background information about space missions, stereo vision obstacle detection and the Lunar Zebro is provided. Whereafter in Chapter 3, the methodology used to conduct this research project is discussed. The requirements of the stereo vision obstacle detection algorithm are generated in Chapter 4. What follows are two design chapters in respectively Chapter 5 and Chapter 6. At first, the design of the new test model is described, after that, the design of the obstacle detection algorithm. Thereafter, the results are provided in Chapter 7, Finally, a conclusion and discussion is condensed in Chapter 8.

2

Space Rovers, Obstacle Detection and the Lunar Zebro

This chapter is to provide the reader with background information about space rovers obstacle detection, and the Lunar Zebro. Most of the content in this chapter has been adapted from the literature study that preceded this thesis [89]. At first, details from previous robotic space missions are discussed, and, in addition, the Lunar Zebro project. Following, a part where computer fundamentals is the main subject, and at last, the challenges of the Lunar Zebro are detailed.

2.1. Previous Robotic Space Missions

In this section, similar projects to the Lunar Zebro are discussed. During each similar case, the main characteristics of that mission or project and the solution employed for obstacle detection are discussed. In the next section, various planetary missions are first discussed; following this, the methods of the Delfy (a flapping wing project of the TU Delft) are elaborated on. A summary of this section is also provided to distil and sum up what has been learned from these missions.

2.1.1. Moon Rovers

The previous rover missions to the Moon have some similar characteristics as the Lunar Zebro. However, in comparison with the Lunar Zebro, the biggest difference found in this small market analysis, is the level of autonomy and scale. All the Moon rovers are discussed in chronological order.

The Space Race Rovers

The Soviet and American space agencies competed in a race toward space and the Moon in the late 1960s and early 1970s. In these missions, they gathered considerable data about the Moon. Together, they explored a small portion of the whole celestial body[41]. During this timeframe, only three sorts of vehicles were developed and deployed on the surface of the Moon: the Lunar Roving Vehicle and the Lunokhod 1 and 2. These rovers are illustrated in respectively Figure 2.1, Figure 2.2 and Figure 2.3. The Lunar Roving Vehicle was crewed and manually controlled, whereas the two Lunokhod rovers were unmanned. Both Lunokhod rovers had been equipped with huge cameras for navigation and scientific research purposes. This footage was sent to the Soviet Union and the rover was controlled manually as well [39]. Accordingly, neither of the space-race rovers contained an autonomous navigation system or any kind of on-board processing.

Chang'e Missions

After almost 40 years of waiting, a Chinese rover set foot on the Moon. The Yutu (Jade Rabbit), depicted in Figure 2.4, was part of the Chang' e 3 mission. After landing on the "Bay of Rainbows" region, this 140-kilo rover was able to drive two lunar days but faced a mechanical problem during the second lunar night. After this, the Yutu rover operated for two more years on the same spot. The rover's objective was to observe and prepare for human exploration by creating a map of the lunar surface. Yutu has a navigation and mapping system that uses a sophisticated stereo vision system [56]. The cameras of the Yutu's navigation system are located about

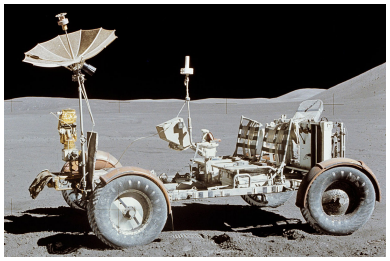


Figure 2.1: The Lunar Roving Vehicle [70]

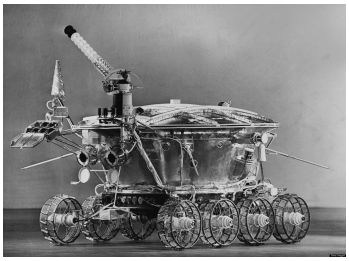


Figure 2.2: The Lunokhod 1 Rover [73]

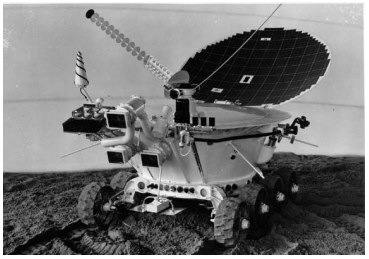


Figure 2.3: The Lunokhod 2 Rover [54]

1.5 m high for as much overview as possible. The viewpoint itself can yaw and pitch to compensate for the camera's limited field of view. To locate itself, the jade rabbit uses a visual odometry algorithm, the data of an Inertial Measurement Unit (IMU) and a wheel odometer. Together they estimate the camera poses.

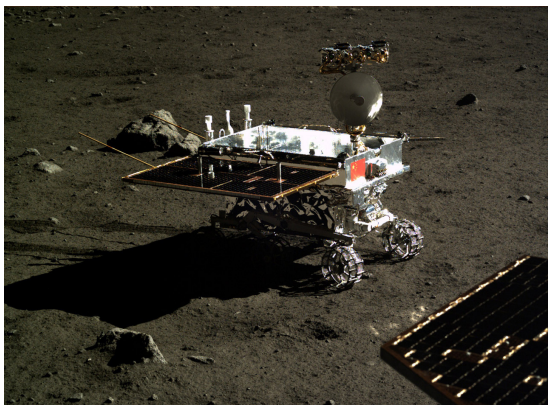


Figure 2.4: The Yutu (Jade Rabbit) Rover [15]

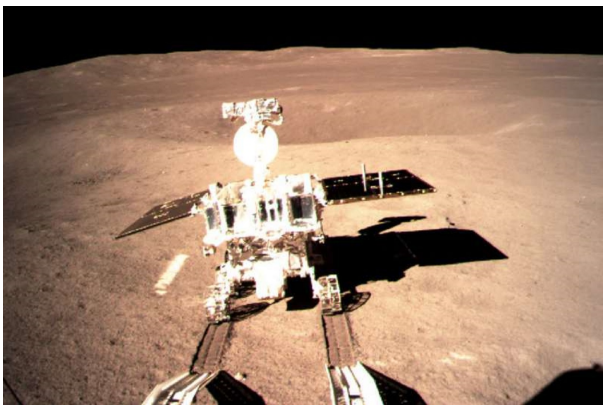


Figure 2.5: The Yutu 2 (Jade Rabbit) Rover [15]

The system was able to plan a pathway autonomously, but it could be corrected by the ground-station operators. To create a map, for path planning and mapping purposes, the rover first takes four to seven shots with the movable cameras in the two free dimensions. The position of the post, however, includes a little inaccuracy. An algorithm tries to match all these images to minimise the misalignment error [56]. Herafter, every picture compensates for the extreme illumination conditions by using a Wallis filter such as the one Tan [103] used. This algorithm seeks to adjust the mean and variance brightness values of areas in the picture so the areas will be in the same range. Before a flight, calibrations, are used to calculate the estimated intrinsic and extrinsic parameters of the camera system. After rectification with the estimated camera model parameters, the stereo matching process will be executed to obtain a depth map of the scene. The used stereo matching algorithm is a semi-global matching algorithm which is based on Hirschmuller[42]. Using the obtained estimated misalignment-errors and taking the four to seven pictures with their depth maps, an overall depth map is now the input of a big computational mapping model. All the depth maps are stitched together using a non-linear minimisation which is based on the Levenberg–Marquardt nonlinear least-squares algorithm [34].

A considerable amount of testing in simulated environments is done prior to the mission for testing the accuracy of the vision system. Moreover, much of the mission data is published online by Zuo et al.[113] and can be used to develop new systems. Currently, there is an ongoing mission on the Moon with another rover, the Chang' e 4 mission. This mission is also deploying a rover that is based on the first Yutu rover. It landed safely on the far side of the Moon on 3 January 2019, according to Jones [45]. The rover is operating, however, not much has been publicised about it yet apart from the photograph (seen in Figure 2.5) and the fact that it is alive.

2.1.2. Mars Rovers

Mars rover missions, which are similar mission types, are examined next. Conditions on Mars differ slightly from conditions on the Moon. However, the rovers are still in unknown terrain. The level of autonomy is important because the downlink to the earth is more and more sparse. NASA has accomplished a couple of

Mars missions. In the next section, these rovers are discussed in chronological order.

Sojourner

On 4 July 1997, the Sojourner[75] became the first rover to ever set foot on Mars. The NASA rover explored Mermaid Dune and was designed as a proof-of-concept mission of a low-cost rover exploration missions to Mars. After it was deployed, the 11 kg rover started its seven-Martian-day mission exploring and conducting scientific inquiry. The lander of the Mars Pathfinder mission was performing its own operations while it was tracking the rover as well. Despite the seven-day goal, the mission continued to operate successfully for almost three months.

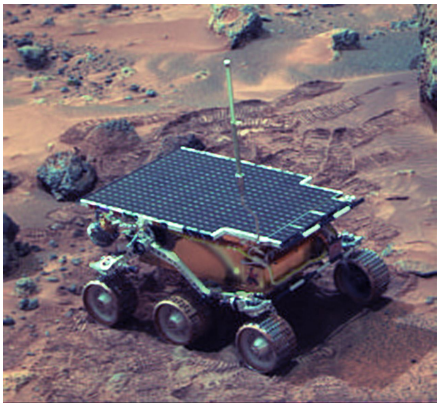


Figure 2.6: The Sojourner Rover [69]

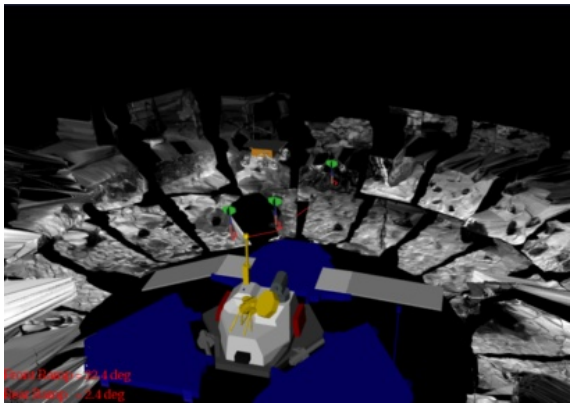


Figure 2.7: Mars Path Finder navigation simulation software [75]

Communication with earth has about an 11-minute delay and can only be established for a few hours per day. Therefore, the rover has a semi-autonomous control system for navigation purposes. From the ground station, the waypoints are determined by software that uses data from the IMP stereo camera system. With this stereo vision data, a 3D map of the surrounding of the lander is created. The operator uses virtual reality glasses to operate a simulated sojourner rover in the software. The determined waypoints are translated to the rover's coordinate system, and transmitted to the rover. A visualisation made by the software is illustrated in Figure 2.7.

When moving, the system stops every 7 cm and senses if there are obstacles in its path. This hazard-avoiding system consists of two camera and five infra-red lasers. These lasers are used for better depth estimation, which is also called structured light stereo vision. With the image data of the patterned scene, a sparse map of the terrain is created. The rover will then try to find the shortest path around the obstacle. After scanning, the rover turns back towards its original heading if the path is still free from obstacles.

Figure 2.8: The Mars Exploration Rover [69]

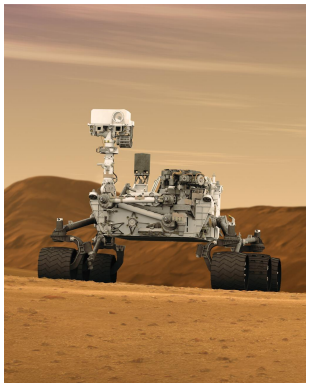


Figure 2.9: The Mars Science Laboratory Rover [69]

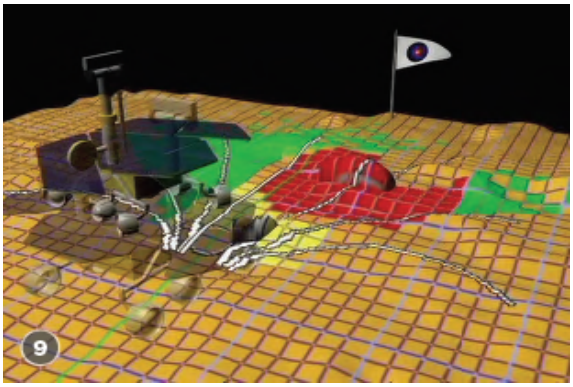


Figure 2.10: GESTALT navigation system assessing the possible paths [69]

Mars Exploration Rovers
The Mars Exploration Rover (MER) mission contained two identical rovers named Spirit and Opportunity. Both set foot on Mars in January 2004 with the main goal of finding the presence of water [64]. The solar-powered MER rovers were not small: they both had a mass of about 174 kg and a size of about 1.5 m by 1.6 m. For navigation purposes, a stereo camera system is installed on a post to provide more overview. Besides the camera system on the mast, there are also some cameras mounted back and front to easily detect unseen hazards. According to Goldberg et al.[37], calibration parameters of these cameras performed on earth will not change significantly, despite all the temperature changes and vibration.

During obstacle detection, images will pass the following pre-processing steps: down-scaling, rectification, and high-pass filtering. After this, disparity computation is done by a dense stereo matching algorithm using 7 x 7 so called Sum of Absolute Differences (SAD) scores. The matches are filtered, and the disparity is further detailed to a sub-pixel level. Uncertain areas are filtered out by a blob filter. All the computations are done in about 30 seconds. A local map is maintained by putting together the depth data, and the localisation data gathered by visual odometry. The local map is divided into a grid. After this, the traversability of the grid is determined using an algorithm which is called Grid-based Estimation of Surface Traversability Applied to Local Terrain (GESTALT) [6]. This algorithm draws arcs on the grid-map and makes a trade-off between the drawn pathways. The pathways of the GESTALT algorithm can be seen in Figure 2.10.

The Mars Science Laboratory mission landed on 6 August 2012. Its scientific goals were to investigate the climate and habitability of Mars. The lander travelled together with a new rover, Curiosity, which is illustrated in Figure 2.9 [53]. The hardware and software of this rover are more or less the same as the MER missions.

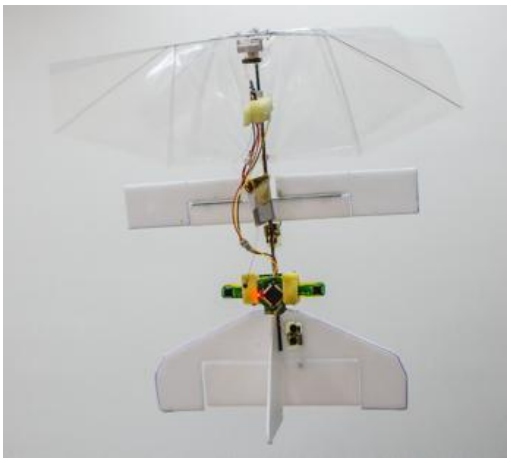


Figure 2.11: The DelFly Explorer [32]

2.1.3. Similar Autonomous Navigation Projects
In terms of their mission requirements, the rover missions appear similar to Lunar Zebro. The hardware requirements however, and especially the scale, differ on some points. In order to gain perspective about the navigation problem on a small scale, the DelFly is investigated in this section.

DelFly
One research group at the TU Delft is studying Flapping-Wing Micro Air Vehicles (FWMAV). Low mass and small scale are characteristics of FWMAV and these are, therefore, inherently safe. The research group called MAVLAB developed the DelFly, which is illustrated in Figure 2.11. As the production of these flying and flapping air vehicles has been successful, research could now focus on the autonomous competences. FWMAV are only able to transport a small amount of load; therefore, there is a significant restriction on sensor weights. Because of these strict requirements, it is challenging to create a real-time autonomous navigation system. This challenge resulted in a 4-gram onboard stereo vision system, developed by Thijmons [105] and implemented in a 20-gram FWMAV—the DelFly Explorer.

The DelFly Explorer has a wingspan of 28 cm, and with 3.5 V it can generate lift and fly around for 10 minutes. Besides the cameras, the Delfly carries an IMU, a magnetometer, a barometer, a communication module, and a motor driver. Computational power on such a low-mass system is limited. Therefore, a new algorithm has been developed by de Wagter et al.[20] named LongSeq, which performs stereo matching on image line level. This algorithm is based on the Semi-Global Matching algorithm of Hirschmuller [42]. To further decrease the computation demands of the system, the designers skip basic processes like rectification and undistortion, and they use sub-sampling to let the LongSeq algorithm skip lines in the picture. Their system is demonstrating computational optimisation; it is not as fail safe as most navigation systems, but it still is showing remarkable results, for example the Delfly achieved corridor traversal [106] a known challenging problem in this field of expertise. The DelFly research group also implemented its developed stereo vision system on pocket drones. With this implementation, they can investigate the navigation of a swarm of drones [65].

2.1.4. Previous Missions Conclusion

While looking at the previous missions, a few conclusions can be drawn. At first, most obstacle detection algorithms use a passive stereo vision system, which follow the same flow, where first images from the scene are pre-processed using filtering, scaling and rectification techniques. Next, pixels in the images are matched. Most of the rovers use the Semi Global Matching algorithm of Hirschmuller for the matching process. Calibration is done before launch and is considered to be consistent over the full lifetime of the rover. A unique challenge for the Lunar Zebro will be the low perspective. All the rovers, including the Delfy, have an overview perspective of the scene. During all the steps, optimisation is possible to make the detection system as light and fast as possible. For example, despite there is a drop in certainty, the DelFly has shown outstanding efficiency in their algorithms. A more detailed market review is shown in Appendix A, where all the currents rovers are listed.

Figure 2.12: RHex-Boston Dynamics [9]

2.2. The Lunar Zebro Project

After looking at other rovers, in this section background information about the Lunar Zebro project is provided by discussing first its anchestor—the Zebro—following by the reason why the Lunar Zebro is created, and its mission goals. After this, a team structure is provide, and finally, the concept and the architecture are discussed.

2.2.1. Zebro

When Gabriel Lopes started working at the TU Delft, he brought the idea of a six-legged robot with him. This bio-inspired six-legged robot called RHex (developed by Boston Dynamics and further developed by Penn State University) is mainly constructed for rough terrain missions [9]. In Figure 2.12, the RHex of Boston Dynamics is visible. Gabriel saw the potential of this bio-inspired robot and designed one himself with the group at the TU Delft. The Zebro (ZEsBenige RObot or in English six-legged robot) became a research project. At first, this research group was housed at the mechanical engineering faculty within the control and simulation

group for studying walking algorithms using Max-Plus algebra [48]. Today, the Zebro is redesigned into the DeciZebro and is used as a platform for research in swarm robotics [81].

Swarm robotics is trying to let robots cooperate in such a way that a group of robots, or a swarm, can perform an overall task or bring about some desired behaviour. The desire to understand swarms arose from observing insects. One individual insect may appear helpless and unintelligent; but a group of insects working together is actually brilliant. A group of ants, for example, excels in searching and locating the optimal trail to a food source, as Deneubourg et al. [21] have described. A swarm does not depend on individuals. According to de Groot [19], if one individual in a swarm falters or dies, a swarm is still able to perform well due to its decentralised organisation. Therefore, scaling up or down the swarm could be applied easily. The performance of the overall swarm will depend on its application. Small robots, compared to large ones, are relatively inexpensive to produce and less complicated. All these characteristics ensure that swarm robotics could be applied to many applications.

Figure 2.13: A walking Zebro [81]

Figure 2.14: A Zebro climbing obstacles [81]

The Zebro is an example of a 'simple' robot that is not overly complex. The Zebro has c-shaped legs which rotate around its axis. When walking, three of the six legs remain on the ground, resulting in a stable base position. The other three legs contribute the walking manoeuvre powered by a servomotor. A hall sensor returns the position of the leg to the locomotion controller. This controller uses the Max-Plus algorithm developed by Gabriels and his team to control the legs [48]. The locomotion system performs a walking gait as illustrated in Figure 2.13. The two angles shown in the picture represent a full revolution. During the walking angle, the Zebro is lifted and placed back on all six feet when the angle is the same as the other legs. Together with the leg-slip, the covered distance can be calculated. In addition to walking, the Zebro can climb obstacles (see Figure 2.14) that are roughly the same height as itself, allowing it to move over many different types of terrain [88].

After several versions of the Zebro, the Deci-Zebro was created by Otten [81], as illustrated in Figure 2.15. Otten aimed to make the design modular. A top-level controller is designed and implemented on a Raspberry Pi, which is an inexpensive microprocessor. This controller handles each of the leg modules and the other subsystems such as a power management system, a communication module and a localisation module. Every system is developed, built and implemented by students. The localisation module includes two ultrasonic sound sensors to avoid obstacles. However, this is still a trial-and-error system since they built some randomness into the design. In this way, the rover theoretically should never get stuck. An actual navigation system has not yet been implemented.

Figure 2.15: Deci-Zebro [81]

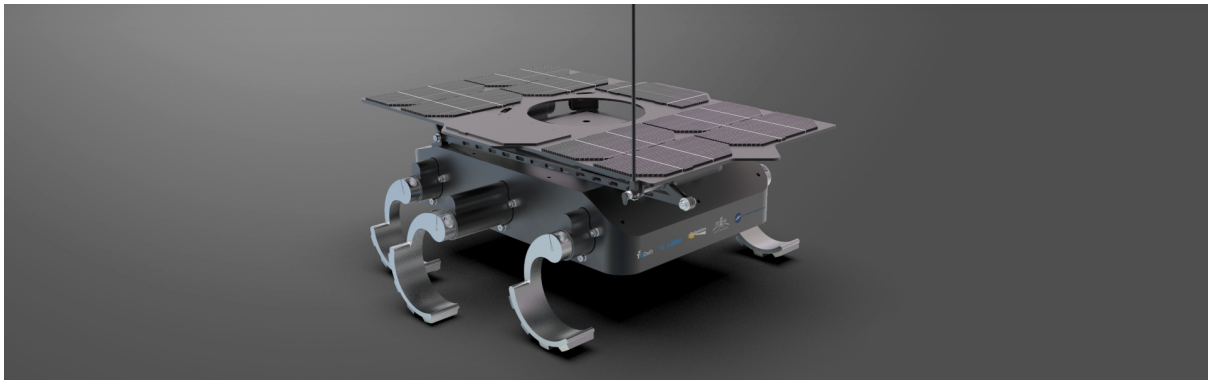


Figure 2.16: The Engineering Model of The Lunar Zebro

2.2.2. Lunar Zebro

The Zebro showed capabilities as a small rough terrain rover and has the potential to be a swarming robot. Therefore, the faculty of Electrical Engineering, Mathematics and Computer Science, together with the Delft Robotics Institute chose to adapt the design to a planetary exploration rover. In light of this new intention, an adaption study has first been carried out, which shows that a rover of this scale might be capable of performing a Moon mission. Sharma's study showed that the c-shaped legs of the Zebro are able to walk on lunar soil [96].

The Lunar Zebro, when successful, would set records as, the lightest, smallest, and first walking rover on the surface of the Moon. The Zebro would provide the TU Delft the unique opportunity to become one of the first universities to visit the surface of the Moon. Nevertheless, why should companies or agencies allow this rover to join their mission? The Simple answer is that this tiny rover could easily fit in. The 'piggybacked' mission would not have to change much of the mission requirements. The company will derive great benefit from including taking the Lunar Zebro in its mission. With this addition, another moving vehicle will be present that can take pictures from both the rover and lander for commercial and scientific purposes.

The team have already developed an engineering model for the hardware of the rover as pictured in Figure 2.16. However, the software for the Lunar Zebro is not yet fully developed. The engineered rover is intended to walk 200 m on the Moon while surviving Moon conditions during one lunar day (which is 14 earth days. During the day, the rover will transmit gathered data and pictures back to earth). Navigation and other commands will be sent back to the rover. The connection with earth will be somewhat discontinuous. Therefore, safe operation requires some level for autonomy of the rover. The operation centre in Dwingeloo in the Netherlands, which is equipped with a single-dish radio telescope, will have contact with the rover for only eight hours a day. This connection will have a low data rate, due to power constraints. The current goal of the Lunar Zebro project team is to develop a working qualification model before August 2021. After this, it could be handed over to a launch partner.

2.2.3. Team Structure

The structure of the team is shown in Figure 2.17. This hierarchy is led by a management layer, where all the professors connected with this project and the chosen department leads are involved; this forms the management team of the project. Management provides the needs and make decisions about the roadmap of the Lunar Zebro. The system engineers transmit these decisions to the engineers, and the chief engineers break it further down into work packages for all the sub-systems. Currently, the team consist of around 40 students.

2.2.4. Lunar Zebro Concept & Architecture

In Figure 2.18, the architecture of the Lunar Zebro is visualised. All the systems listed are the main electronic components connected by a carrier board. The structural component, which also plays a significant role, is not visualised here.

The locomotion system, designed by Rouwen [88] and Miog [67], is based on the Deci-Zebro and has been successfully implemented in the engineering model. A space qualified brush-less motor replaces the servo

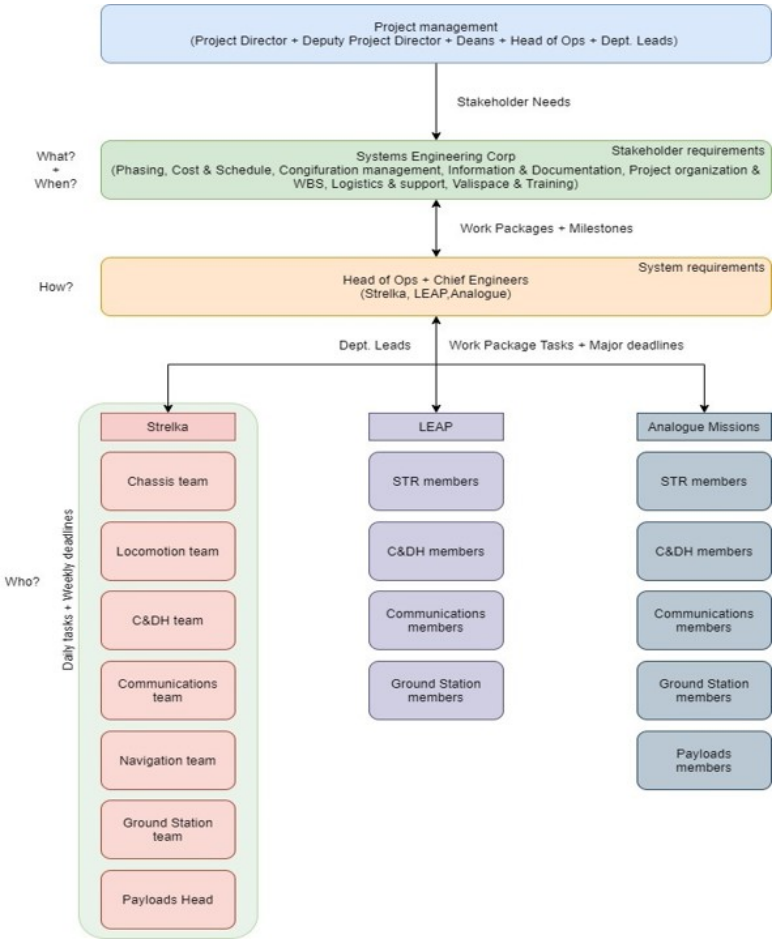


Figure 2.17: The Structure of the Lunar Zebro Project

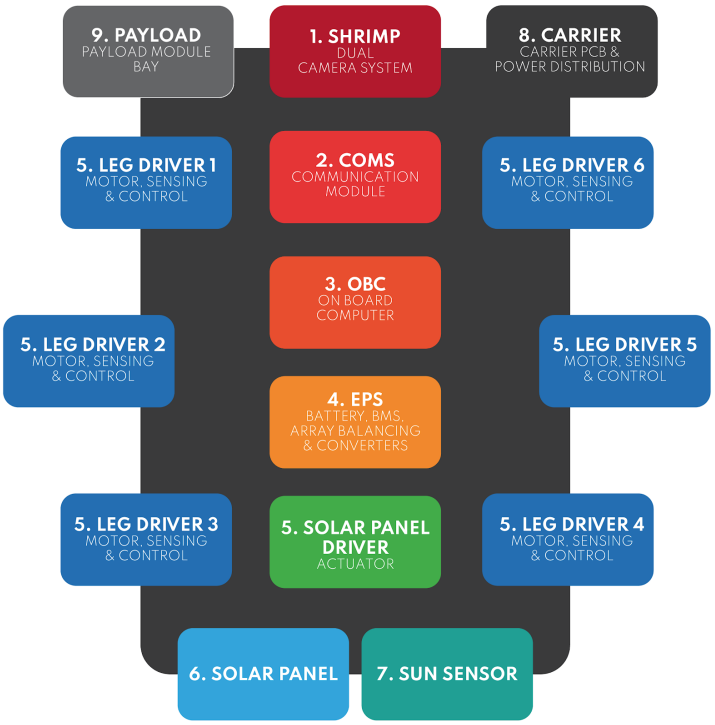


Figure 2.18: The Architecture of the Lunar Zebro

motor, and the communication between the On-Board Computer (OBC) and the motor drivers goes through an RS422 full-duplex bus instead of the I2C bus in Otten's design. TID Radiation testing has been performed up to 20 krad; each of the individual leg-drivers is protected against overvoltage, overcurrent and latch-up situations.

The CP400.85 onboard computer developed by Hyperion [68] is a processing unit with a CPU speed of 500 MHz and low energy consumption. The size of the CP400.85 is small—only 20 x 50 mm. The operating software on this processing board is based on Linux, which makes it very flexible. It has 512 MB of allocated RAM, and it can host up to two SD cards. The onboard computer or Zebro Processing Unit (ZPU) runs a master control program called TRON. This control program makes the rover walk and navigate on the Moon by controlling all sub-processes like obstacle detection system.

As seen in Figure 2.18, a stereo vision camera system is implemented on the Lunar Zebro. The used camera is called SHRIMP. In Section 4, the SHRIMP camera is further detailed.

2.3. Computer Vision Fundamentals

As seen in the previous section, the used technique for acquiring 3D information from the scene in the Lunar Zebro, and also in many other previous missions, is passive stereo computer vision. Humans can sense the world around them very easily. With just one look, they can observe in great detail the scene in front of them. With the help of shapes and lighting, the human mind can recognise objects and isolate them from the scene. However, the mechanism behind such a 'simple' process is actually quite complicated. Where cognitive psychologists try to understand the inherent optical advantage of a human, computer vision researchers try to imitate the natural process with mathematical calculations. In recent years, techniques to abstract data from a picture—such that a computer perceive the three-dimensional surrounding—have been improved. The visual variety is so great that it is almost impossible to create a perfect model. Geometry, surface microstructure, illumination, colours, reflectance or light scattering are the problems a vision system has to deal with. Humans do this instantly without any effort, whereas computers have to work hard to process all the information. However, computer scientists are gradually mastering this area with greater precision. More sophisticated computer vision in many applications in real-life is already reflecting their efforts. A suitable algorithm has to be efficient and robust to noise and deviation [102]. In this section, an overview of the theory behind computer vision is given. First, the perception of the scene is discussed, followed by a description of the camera model. This camera model is used to calibrate the stereo geometry. Because with this stereo setup, depth can be obtained and finally, obstacles in the scene can be detected.

2.3.1. Perception

Acquiring perception is one of the critical capabilities of an obstacle detection system. Perception is basically the sensor information about the environment. There are multiple ways to range or sense the scene around the robot. Multiple options are discussed in the literature review that preceded this thesis [89]. Sensors use normal reflected light, active lighting, sound, or even electromagnetic radiation. This thesis focus only on a passive stereo vision system. This system is also installed on the Lunar Zebro and is widely used among the other planetary rovers discussed in Section 2.1.4.

Passive light sensors catch reflected light from objects in the scene and storing it on an image plane. This technique is already ancient: the first photograph was made almost 200 years ago by a Frenchman named Nicéphore Niépce using the camera obscura technique. The camera obscura is a dark room with one small open hole to the outside world. Niépce made a portable version which is called a pin-hole camera. In addition, Niépce used a light-sensitive film, a paper coated with silver chloride [80], to catch the light entering the pin-hole camera. Today, the pin-hole camera is optimised with lenses, mirrors and shutters, and the photographic film is replaced with digital light-sensitive sensor arrays. The output of this data is processed by mechanical elements, electrical elements, and computers. A typical architecture of a modern camera is shown in Figure 2.20.

2.3.2. Camera model

The rays of reflected light from the scene enter the camera through a lens or multiple lenses, which will focus the rays on a light-sensitive sensor as described in the previous section. Two physical parts of the camera—the

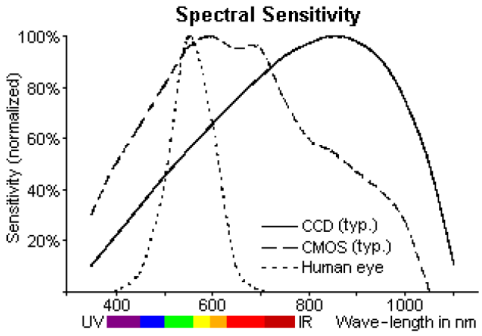


Figure 2.19: The spectral range of a human, a CCD sensor and a CMOS sensor [35]

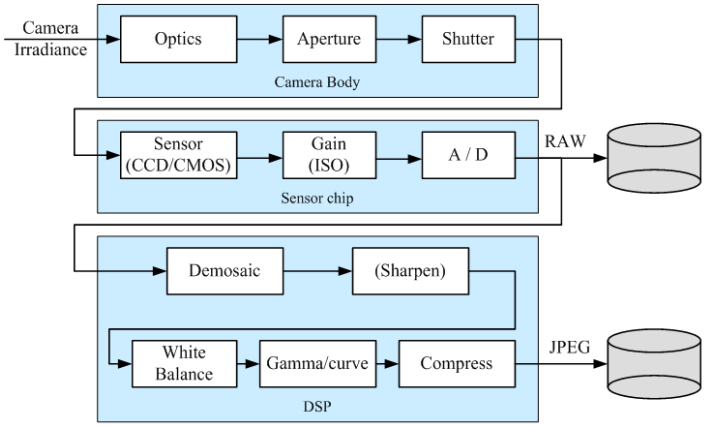


Figure 2.20: A representation of a needed architecture to take a picture [102]

aperture and the shutter—influence the amount of light that is finally sensed. Together with the lens, these components influence the image brightness, motion blur, field of view, and depth of field[35].

There are two versions of light sensors available: a Complementary Metal Oxide Semiconductor (CMOS) and a Charge-Coupled Device (CCD) sensor. For obtaining a RGB picture, a colour filter with a Bayer pattern is typically used to pass on one of the three colours on one pixel. Both sensors have a different spectral range, which lets the sensors catch more than just visible light. Both spectral ranges are visible in Figure 2.19 where the spectral range of the human eye is highlighted.

The most important component of a camera system is the lens. If the lens were perfect, the rays would go straight through the lens and be perfectly deflected. However, in reality, seven aberrations can occur. They are divided into two groups: monochromatic and chromatic aberrations. Chromatic aberrations follow from the mechanical properties of the lens. Not all wavelengths have the same refractive index and therefore are deflected differently, which could deviate in both axial and transversal directions. Monochromatic aberrations are an umbrella word for multiple aberrations, namely defocusing aberration, spherical aberration, comatic aberration, astigmatism, field curvature, and image distortion [79]. The biggest aberration for computer vision purposes is radial distortion. Wide-angle lenses make straight lines appear to be curved in the image. Moreover, the reconstruction of the scene becomes more complicated. Fortunately, this distortion is easy to compensate for in practice. An illustration of radial distortion is shown in Figure 2.21.

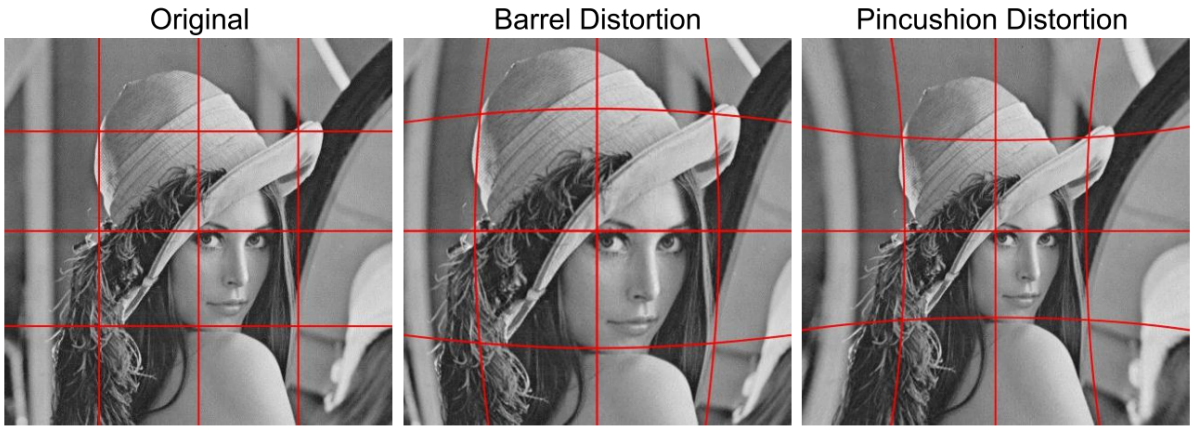


Figure 2.21: An example of radial distortion [47].

A camera model is needed to describe u, v coordinates, which are the image plane coordinates, to x, y coordinates in the real world. This model includes extrinsic parameters and intrinsic parameters, also shown in

Figure 2.22. The extrinsic parameters are the Euclidean transformations between the camera and the world coordinate system. The intrinsic parameters, on the other hand, represent the transformation from the camera-to-pixel coordinates. The intrinsic parameters include the focal length, projection of the optical centre, the aspect ratio, and the skewness; it could also include the lens distortion parameters. In order to obtain these parameters, a calibration algorithm could be used. For example, Zhang describes a calibration method[112] which could estimate these extrinsic and intrinsic parameters.

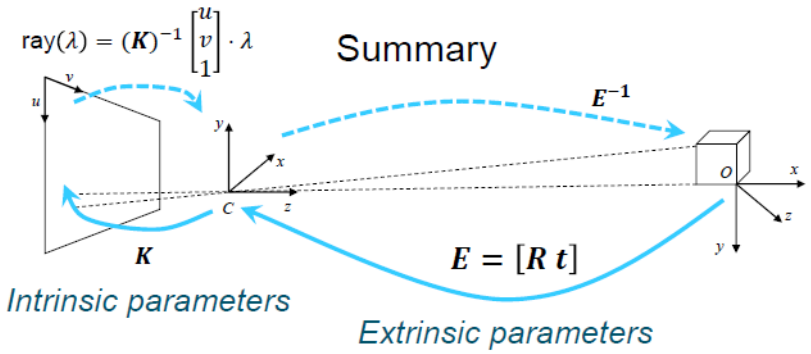


Figure 2.22: A projection model from the world reference frame to the image plane and vice versa [51].

2.3.3. Stereo Vision

Stereo vision gives a system the ability to estimate depth with high resolution in the scene. One benefit of using a stereo vision system rather than another range-sensing system is that it does not require actuation or any additional external light source. This benefit results in a lower total mass. Stereo vision works in many different lighting conditions and is therefore applicable in many environments. This technique is already applied in many different fields, and considerable development is being achieved.

2.3.4. Stereo Setup

How does a stereo system work? As seen in Figure 2.24, two different perspectives are obtained through the shift in one direction of the two cameras. This shift could be in any direction on the epipolar line. An epipolar line results from the fact that every point in one image represents a line in the other image. This phenomenon is illustrated in Figure 2.23. If one sees the X_L in one image, the real point X could be in X, X_1, X_2, and so on. All these points are forming a line, which is the epipolar line. When applying this to multiple points in the left image, the epipolar lines will cross each other in one point called the Epipole e_r. This relation between the two camera could be mathematically described using two matrices—the essential matrix and the fundamental matrix.

The essential matrix describes the relationship between the two image planes of the two cameras and contains a rotation and a translation matrix. The fundamental matrix uses this essential matrix together with the two intrinsic parameters of both camera systems. A stereo calibration method could help to find the fundamental matrix of the stereo setup. With the fundamental matrix, the stereo setup could be rectified. This will force the epipolar lines to be horizontal. A point in one image will now be on the same horizontal line as its match in the other image. This will have a great computational benefit for stereo matching algorithms[35].

2.3.5. Stereo Matching

Stereo matching is crucial to determine the shift of objects within an image pair. The shift of an object in the pictures gives information about the depth. The more it shifts, the closer the object is to the cameras. This shift is called disparity. There are two methods for obtaining the stereo-corresponding image points to calculate the disparities of a scene: dense and sparse stereo matching.

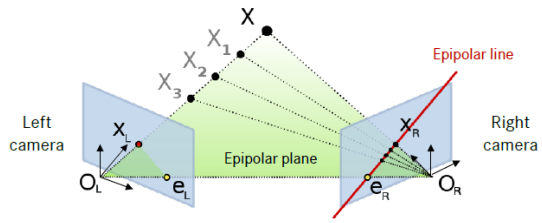


Figure 2.23: A stereo setup with epipoles and epipolar lines illustrated [35].

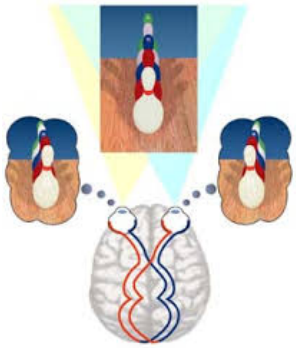


Figure 2.24: Disparity through human eyes [16]

The first option is dense stereo matching. Dense stereo matching algorithms process all the pixels in an image. Each pixel is matched to pixels in the scan line in the other image using a squared window around it, called a kernel. The kernels of each pixel are matched using similarity measurement methods such as Normalised Cross-Correlation (NCC), Sum of Squared Differences (SSD), Sum of Absolute Differences (SAD), or Semi-Global Matching (SGM) or other similarity measurement methods [17]. The performance of all these different algorithms is compared by multiple frameworks like the Middlebury website [18], and the Modular Stereo Vision Framework [77]. Difficulties during stereo matching are occlusions, specular reflections, homogeneous regions and repetitive patterns. Occlusions are regions that are only kept by one camera instead of two. Specular reflections are the mirror-like reflection. Homogeneous regions are regions with not enough texture difference. Repetitive patterns are difficult to match because of their repetition.

All these algorithms contain numerous changeable parameters to influence the matching performance. The most important parameter among them is the window size. Changing the window size of the kernel will result in more fluent disparity, but the accuracy will change. In Figure 2.25, this kernel change is visualised. As depicted, a large window provides reliable matches; however, small window sizes provide more accurate matches. So, an optimum between reliability and accuracy needs to be found, which is different in every case.

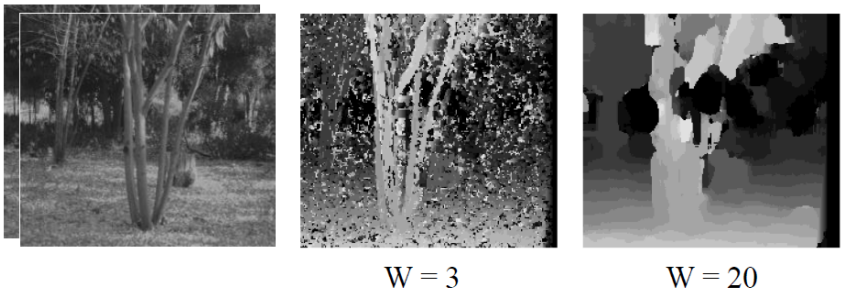


Figure 2.25: Two different disparity map results of the same matching algorithm using two different window sizes. [26].

The other method is called sparse matching, which only uses feature matching. Features contain bits of information of useful image parts. Features could, for example, be edges, corners or structures. First, an algorithm looks for a feature within an image. It creates a descriptor of the feature and matches the descriptors of the two images. These features could be scale, rotation and viewpoint invariant. The algorithm could, for example, skip the rectification process. Feature matching is a method which is robust for occlusions, noise, and illumination differences. A widely used method for detecting and matching features is Scale-Invariant Feature Transform (SIFT)[111]. Feature matching is used in many solutions such as image alignment, motion tracking, object recognition, robot navigation and stereo reconstruction. The greatest downside for using sparse stereo matching is the lack of performance in a monotone landscape, which has a lack of texture.

2.3.6. Rectification

Rectification can significantly reduce the computational intensity of dense stereo matching, since it can now be assumed that every pixel in one image is in the same row as the other. In Figure 2.26, the rectification

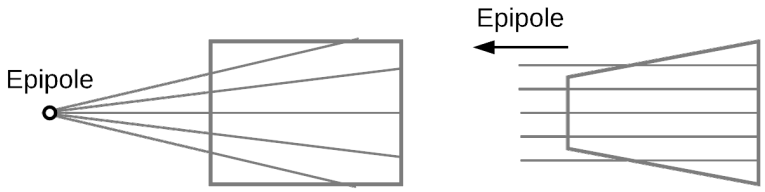


Figure 2.26: Epipolarlines and Epipoles before and after Rectification.

process is visualised. It is clearly seen that the epipoles are manipulated to be in infinity. This manipulation is done by nothing more than the application of projective transformations to images with predetermined stereo setup parameters as the essential matrix and the fundamental matrix. The fundamental matrix $\overline{\mathbf{F}}$ for a stereo setup with a predetermined epipole $i = \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$ will be represented as follows:

$$\overline{\mathbf{F}} = [\mathbf{i}]_x = \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & -1\\ 0 & 1 & 0\end{bmatrix}$$

The homographies $\mathbf{H}_R^T$ and $\mathbf{H}_L$ describe the image projection for the left and right image needed for rectifying. Together with the basic formula for determining the fundamental matrix, both homographies together with the $[\mathbf{i}]_x$ should meet the Formula 2.1. The fundamental matrix $\mathbf{F}$ is now a function of both image transformation matrices. Loop et al. [58] describe a method for obtaining these homographies such that image distortion due to image projections is reduced.

$$\overline{\mathbf{u}}_R = \mathbf{H}_R\mathbf{u}_R \;\&\; \overline{\mathbf{u}}_L = \mathbf{H}_L\mathbf{u}_L$$

$$\overline{\mathbf{u}}_R^T\,\overline{\mathbf{F}}\,\overline{\mathbf{u}}_L = 0 \rightarrow \mathbf{u}_R^T\underbrace{\mathbf{H}_R^T[\mathbf{i}]_x\mathbf{H}_L}_{\mathbf{F}}\mathbf{u}_L = 0$$

$$\mathbf{F} = \mathbf{H}_R^T[\mathbf{i}]_x\mathbf{H}_L \tag{2.1}$$

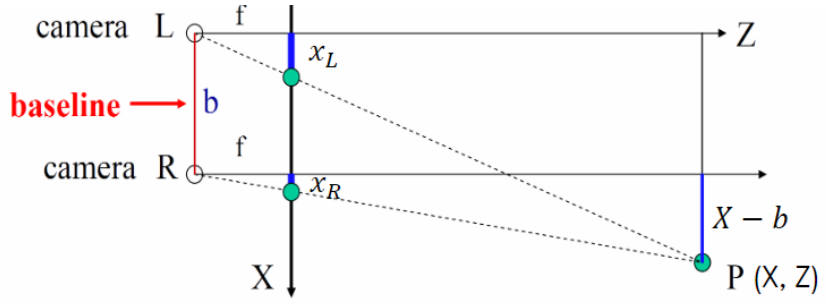


Figure 2.27: A Stereo Camera Setup [35].

2.3.7. 3D Reconstruction

After defining disparity, depth could be defined. Calculating depth is done by a method called triangulation, which basically means translating the shift in the images to real 3D coordinates. By knowing the disparity values from the stereo matching process and the camera's parameters for the calibration process, the depth could be calculated using the following Formula 2.2.

$$P_z = f\frac{b}{x_L - x_R} \tag{2.2}$$

P_z represents the depth in z-direction of point P, f the focal length, and b the baseline of the stereo setup. This formula is derived from the triangle formed by the camera setup. The triangle is shown in Figure 2.27. The disparity $(x_L - x_R)$ is, according to the formula, inversely proportional to the depth. In theory the formula

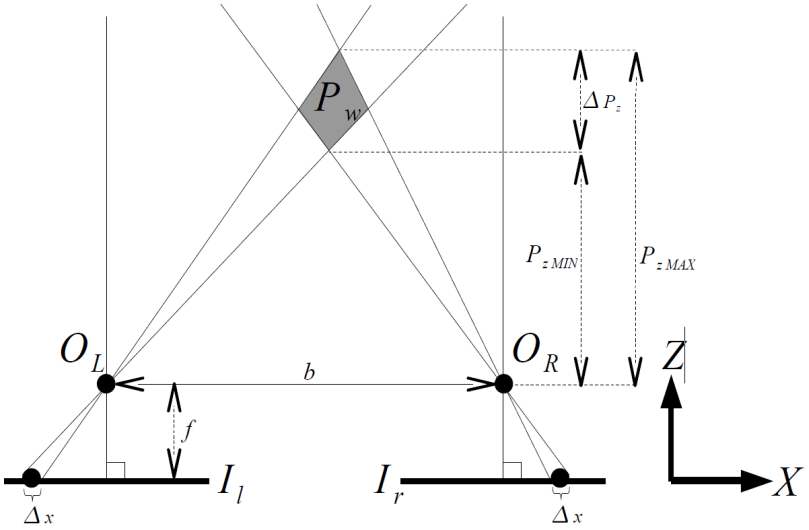


Figure 2.28: Uncertainty while calibrating

in Equation 2.2 should work perfectly. Nonetheless, due to discretisation, an error is occurring. The simple fact that the camera is using pixels to capture the scene makes this perfect model not perfect anymore. Every point has a certain width; when accumulating this over the distance the real depth suddenly becomes a little surface instead of one point. The distance is therefore an estimation. Dubbeldam [23] derived an equation from using Figure 2.28 and the triangulation Formula 2.2. This formula describes the relationship between the error in the depth ΔP_z and the distance of point P in z direction, P_z. Furthermore, the focal length of the cameras f, the distance between the cameras b, and the pixel width Δx is used.

$$\Delta P_z = \frac{\sqrt{2}\Delta x p_z^2}{fb} \tag{2.3}$$

Besides discretisation of the camera's image plane, more errors are sneaking in. The calibration process also has a re-projection error, and the matching algorithm disparity estimation is also not perfect. A final error needs to be calculated using the final used stereo setup parameters and the used algorithms. This calculation will happen in Section 7.3 where this final error is elaborated on.

2.4. Lunar Zebro's Challenges

In this section, specific challenges of using computer vision on the surface of the moon are summed up. Both environmental challenges as well as the Lunar Zebro challenges are elaborated.

The surface of the Moon consist of many monotone landscapes and obstacles. This landscape is visualised in Figures 2.29 and 2.30, where craters and rocks could be seen. While capturing images on the moon illumination difficulties arise when the sun has a low inclination angle to the lunar surface. Those difficulties will cause two effects, at first, it forms long and harsh shadows. These harsh shadows will also cause varying temperatures, affecting the image quality. Furthermore, the sun also forms a thread when shined directly on the lens causing solar flares or over-lightening. Additionally, dust could stick to the lens due to its electrostatic charge [41].

For the Lunar Zebro, the limited hardware is based on the low mass requirements. Every bit of additional required power will result in larger batteries and more solar panel area. Thus, the camera and On-Board Computer will not be designed to produce high end quality photos, but optimised with mass and power restrictions.

Finally, as previously discussed, the low perspective will make navigating the Moon's surface very challenging, because the stereo vision detection system is lacking overview.

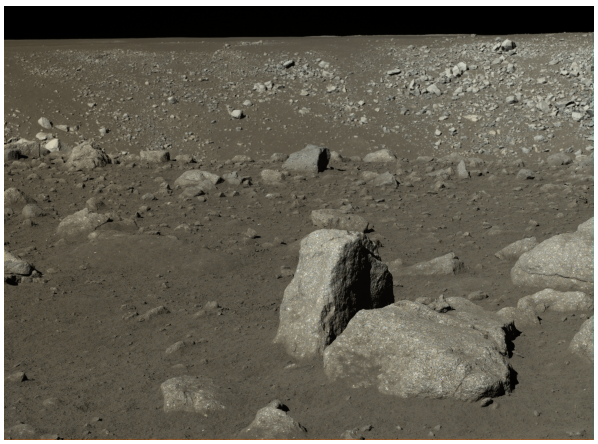


Figure 2.29: Rocks on the surface of the Moon [60]

Figure 2.30: Rocks and craters on the surface of the Moon [60]

3

Methodology

This methodology chapter provides insight into the steps needed to conduct the research. In the first section, the development plan of the thesis is discussed and its effect on the conducted steps. Following this is the scope where all the conducted steps are framed, and how the requirements were set. In the third section, the experimental set-up is discussed. After this the software development is elaborated on and the last section discuss the validation method used to validate the compliance of some requirements.

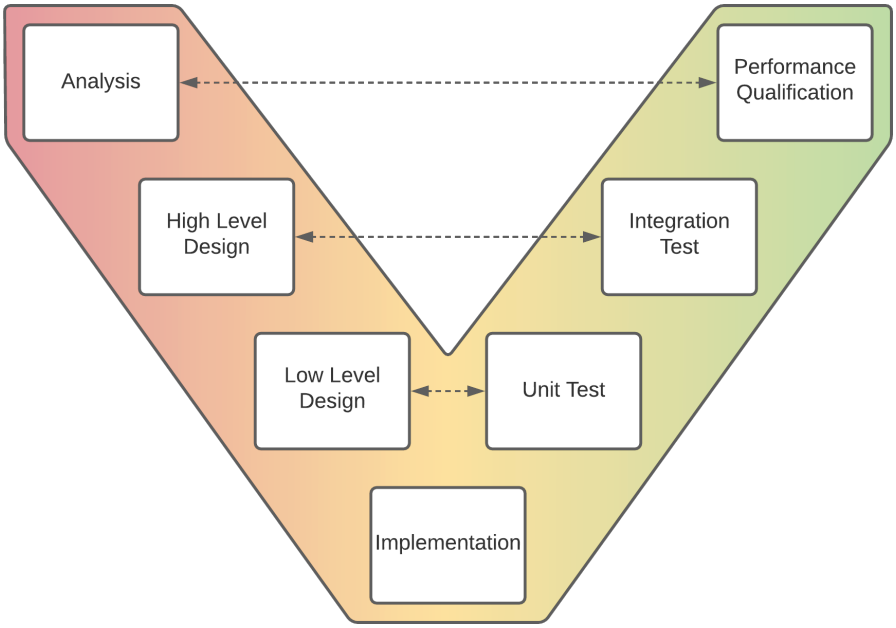


Figure 3.1: V-model used as development method

3.1. Development Method

The method used for developing OPAL follows the V-model in Figure 3.1. The flow of this model from problem to result is visualised by the colours red to green. The V-model represents a dynamic approach of a design cycle by accepting changes if later stages reveal the need. The first block in the chain is the analysis block. The analysis phase produces a clear picture of what is required to have a successful end product by setting requirements and making a development plan. In the last block, this analysis is tested using this set of requirements. Next, a high-level design is developed. This high-level design contains an architecture that outlines which modules

are developed and how these modules will cooperate. This flow of modules is tested within the integration test block on the V-model. The next step is the low-level design, that is containing the development of these modules. Every module will be evolved so that every small unit will work on its own. After implementation, all these separate units are tested in small chunks. Finally, when the end product is validated the development is finished. The V-model is translated to steps which are conducted in this thesis. When looking at the steps actually two little V-models could be spotted.

1. Define Requirements
2. Develop and Build a Test Model
3. Record Data in a Moon-like Environment
4. Develop an Algorithm
5. Process the Data
6. Validate Detection and Distances

3.2. Scope

These steps follow from the research objective, which is framing the scope of this thesis. This research objective is stated as: Develop an obstacle detection system for the Lunar Zebro that will detect rocks and their relative distance in the lunar environment. The challenging characteristic of the rover is the perspective. From Chapter 2, where it is known that all the previous rovers were equipped with a high placed camera system. The data produced by these large rovers will not represent the problems the small Lunar Zebro will face. Therefore the second goal of this thesis is set to: gather data in a lunar-like environment that will be a representative input for OPAL. Furthermore, the resources of the Lunar Zebro project lead to additional restrictions. The project depends on students and companies who have developed both the software and hardware of the Lunar Zebro. When this thesis started, both the SHRIMP cameras and on-board computer were not available. A test model to test the stereo vision obstacle-detection system needed to be developed. Hence, this became the third goal of the thesis: Develop a test model that simulates the SHRIMP camera's behaviour and characteristics, the on-board computer, and the Lunar Zebro itself.

There are a couple more restrictions that need to be taken into account. First, this thesis is subject to limited time, which is a significant driver limiting the scope of activities. For example, the development of the algorithm is primarily performed in Python. The reason for this is to reduce the learning curve needed to get familiar with C++, which is the required programming languages for implementation with TRON.

OPAL is focused only on the working principles of the stereo vision detection algorithm. Risks like launch vibrations, lunar radiation, communication issues, power issues, effects of extreme temperatures, shadow impact, optic illusions and craters were not mitigated in this thesis.

Where the scope is bounding the conducted work, the requirements are, as discussed, specifying features and functions with which the result must comply on. Since this is a pioneer study, the development is brought to an end and is seen as the first iteration. Much tweaking, optimizing, and designing is required to make OPAL ready for a lunar mission. In Chapter 4.2 the interface and the performance requirements are set to a point of readiness that is considered to be satisfying this first iteration.

3.3. Experimental Set-up

The experimental setup consist of actual or close to actual hardware and a simulated environment. The design of this test hardware is discussed in Chapter 5. With this test hardware, tests are conducted to gather stereo vision data and position data.

These tests took place at Decos, a software company with its main office in Noordwijk. The CEO of Decos, Paul Veger, is a space enthusiast whose office location looks as though a meteorite dropped on a Mars-like surface. This building is depicted in Figure 3.2. The yard around the building resembles the surface of Mars.

Figure 3.2: Office of Decos in Noordwijk

To evoke this resemblance, numerous rocks, a grainy surface, and crates have been used.

This Mars-like surface is unique in the Netherlands, since the number of analogue test locations is severe. This Mars-like yard does have similarities with the lunar surface, like the types of rocks and the grainy surface. Therefore, testing on this surface is a unique chance for the test rover to mature itself while gathering sufficient data. Details of the test setup are enclosed in Appendix B.

During the development of the test platform in Chapter 5 everything was prepared, so the operations on the test day itself were straightforward and easy to carry out. On the day itself multiple test cases where carried out. Most of them were walking straight towards obstacles, but other test cases like passing rocks or walk in between rocks were carried out as well. The calibration of the stereo setup was tested too. Before almost every video, the system was calibrated first by shooting footage of a chequerboard, to test whether the setup is changing over time. The results are stated in Chapter 7.

Table 3.1: Used software for the development of OPAL

Software Tool	Version	Purpose
OpenCV	4.1	Software library for image and video processing
Python	3.9.1	Programming language that OPAL is written in.
Matlab	R2019b	Mainly used for the Stereo Calibration Application integrated in Matlab
Visual Studio Code	1.53	Used IDE for code developing and debugging
Git	2.31.1	Software used for Version Control
GitLab CI/CD	13.0	Tool used for running code after code changes
Scikit-learn	0.24.2	Python library used for useful data processing algorithms.

3.4. Software Development

Since it is a single case study acting as a setup for more detailed research in optimising this system, significant parts of the algorithm originate from open-source libraries. The literature will primarily form the basis on which algorithms are selected from this extensive set of commonly used algorithms. Connecting or integrating all the systems into one large algorithm required understanding of the used algorithms, and sometimes this required additional adaptions.

In Table 3.1, the tools used while developing the code are listed. For example, as noted in this table, programming languages, used libraries, and used coding tools are present. A GitLab Continuous Integration,

Continuous Delivery and Continuous Deployment (CI/CD) method is used for running the algorithms. On an Intel® NUC Board (NUC5i5MYBE), a GitLab runner (an application that runs CI/CD pipelines) is installed. Every time a significant code change is pushed to Git (a version control system) a new pipeline is created and run by the GitLab runner. The output is stored on this Inter NUC server.

3.5. Validation

The system requirements are validated using four different methods. First, there is the inspection method. In this method, the end product is examined with only basic senses—as for example, when inspecting whether a particular button is existing. The second method is the analysis method. A person could perform some calculations or predictions in the analysis method to validate whether the product complies with the requirements. The third is the demonstration method to see if the product is complying with the requirement to perform as it is intended. Finally, the test method is a little more refined than the demonstration method, as in this method, for example, specific behaviour is desired.

The final results are ran using the development platform as discussed in the previous Section 3.4. The videos of the test cases are stored locally on the server and processed by the developed algorithm. The results only need to be validated. This validation process is based on a known performance parameter known as Average Precision (AP). AP is, namely, a widely used parameter to identify the performance of an object detection system [82] [38] [28]. To calculate this AP, additional clarity on more parameters needs to be provided. These parameters are:

1. True Positive (TP), which indicate a 'True' detection of the ground-truth obstacle.

2. False Positive (FP), which indicate a 'False' detection of the ground-truth obstacle—the detection is misplaced, or it is detecting an absent obstacle.

3. False Negative (FN), which indicate a 'False' non-detection, while there is an obstacle.

There is a fourth similar parameter which is called the True Negative (TN). The TN parameter is, however, not traceable since the number of possible bounding boxes, that could be true negative in a single picture, is theoretically finite but practicality infinite. Scoring a positive detection is done with an overlap ratio, which is called Intersection Over Union (IOU). The IOU defines the overlap between the detected obstacle bounding-box and the true obstacle bounding-box over the area of union. This formula is visualised in Figure 3.3.

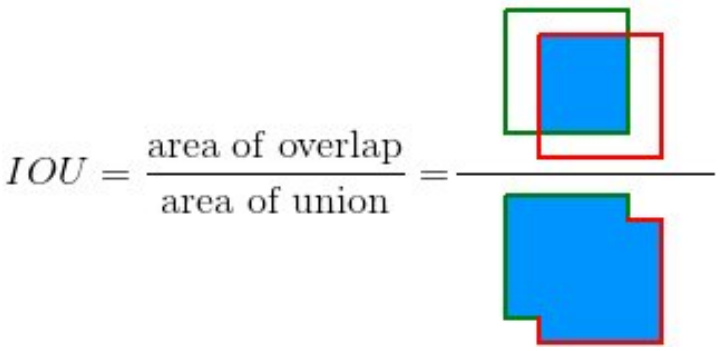


Figure 3.3: Visualisation of the Intersection Over Union (IOU) [82]

If the TP and FP are determined, then the Precision (P) and the Recall (R) could be established. The Precision is the chance a detected box corresponds to a ground-truth bounding box, although the recall signifies the chance that all bounding boxes are detected. Notice that, while aiming for high precision, the recall will automatically drop and vice versa.

$$P = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{TP}}{\text{all detections}} \tag{3.1}$$

$$R = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\text{all real obstacles}} \tag{3.2}$$

Most object detection systems use neural networks that output for multiple classes a certainty parameter. This factor is referred to as called confidence. A confidence threshold blocks the object with low confidence and only shows the bounding boxes with high confidence. When varying with this confidence threshold, a PR curve could be made. The area beneath this area would be the average precision of the obstacle-detection system. However, the designed algorithm does not use machine learning because of resource restrictions and it is not outputting confidence levels. Therefore, the output of the model will consist entirely of the Precision and the Recall. For now these parameters are used as an indication whether the algorithm is working sufficiently, in a later stage these parameters could be used for tweaking an optimisation.

The data set, created during the Decos Test, contains only images. For performance evaluation of the algorithm, ground-truth data is needed. Determining these could be done manually or by a computer, specifically by a Region Of Interest (ROI) tracker. The CSRT tracker of Lukezic et al. [59] is considered a good choice for tracking this manually defined ROIs. However, the CSRT experiences difficulty when the object is occluded, for example, behind another object. Since the CSRT could achieve high framerates, the process can easily be monitored. It results in a semi-automatic tracking system, with the human in the loop to minimize tracking errors.

So with this ground-truth definition, for every result, it could be checked if it is either a true or a false positive. Furthermore, the algorithm is outputting for every detected obstacle a measured distance. With measurements using a measuring tape, the ground-truth distance is known. Now, the test cases can be visualised in one graph. The distance is estimated using a constant speed in between the measurement points since no real-time distance was available. The reason for this is elaborated in Section B, which is also the reason why only a few of the test cases were valid to use. The algorithm's overall performance in these test cases is visualised in the final result graph, where precision and recall are included. Hereafter, with this graph, a conclusion can be drawn whether OPAL is meeting the requirements or not.

4

Requirement Generation

Requirements are the pillars of every engineering project. Without knowing the requirements of the system, there is a risk of building useless or inadequate products. In particular, when determining whether a product is final or good enough, a clear definition of the end result helps one know what and how to test. Additionally, a clear definition of requirements will help this thesis substantiate the choices made during to period.

The algorithm performing the stereo vision obstacle detection is also called OPAL, an acronym for Obstacle Processing ALgorithm. To clearly define all the requirements for developing OPAL, the system's interfaces are first explained. Next, a functional analysis is conducted, and the origin of the key requirements are further discussed. In the end, all these requirements will form a complete list of requirements. This list is a document that will change over the course of the project and is, therefore, a dynamic document.

4.1. Interface Analysis

At first, an interface analysis is used to provide a clear picture of what could influence OPAL, which specifications OPAL should meet, and what the system's inputs and outputs are. Other apps or hardware specs could facilitate or complicate OPAL's output at a later stage of the design cycle. With the N2 Chart, these influences are monitored. In this section, the N2 Chart of Lunar Zebro's navigation system is first illustrated. All these interfaces are then elaborated on.

4.1.1. N2 Chart

One of the tools widely used during system engineering is the N2 Chart, which could help visualise the systems inputs and outputs. This tool is also usefull in identifying natural clusters. One reads an N2 Chart clockwise. All the sub-systems are on a diagonal line. The cells on the right and left side on the same row as OPAL in the N2 Chart are OPAL outputs to other sub-systems. The cells in the same column as OPAL are the inputs driven by the other sub-systems. The N2 Chart in Table 4.1 involves all the components of the navigation system of the Lunar Zebro. Clearly, because of the Lunar Zebro and TRON's modular architecture, OPAL is

SHRIMP	Pictures and state data upon request			
Requests for pictures and state data	**TRON**	Pictures and commands		
	Closest obstacle and statedata	**OPAL**		
Power to SHRIMP	Power to OBC, TRON runs on OBC.		**EPS**	
Connected to structure at the front of the Zebro				**Body**

Table 4.1: N2 Chart of the Navigation System of the Lunar Zebro

only interacting with TRON. TRON is the name of the software running on the On-Board Computer (OBC). This OBC gets power of the Electrical Power Subsystem (EPS) and the body house it. Furthermore, TRON takes care of the interaction with SHRIMP, which is the name of the camera module. Therefore, OPAL does not have to handle timing and communication issues. OPAL is provided only with TRON's pictures and commands, and it will solely return the closest obstacle and status data. Together TRON and OPAL form their own 'circle'.

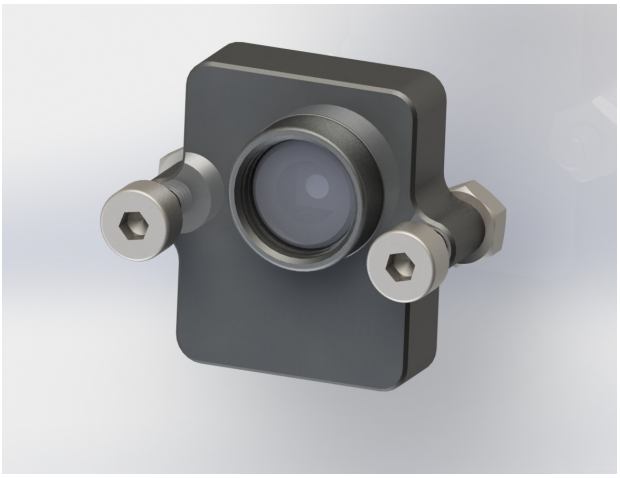


Figure 4.1: The SHRIMP module [1]

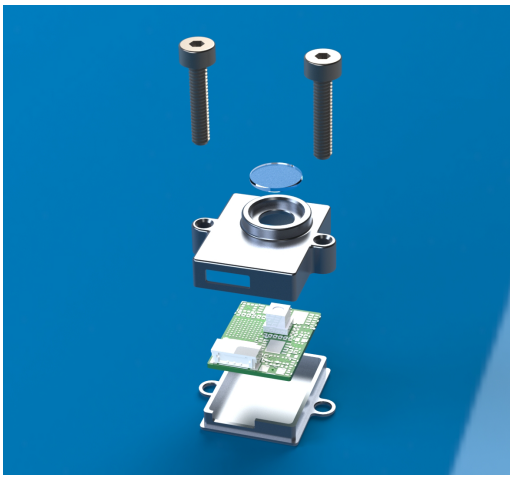


Figure 4.2: The SHRIMP module exploded view [1]

4.1.2. SHRIMP

Besides SHRIMP not being a direct interface to OPAL, it will have a major influence on the performance of OPAL. SHRIMP claims to be the smallest (15.5 mm x 12 mm x 5.6 mm) and lightest (< 3 g) space-graded camera developed. The camera consists entirely of Commercial Off-The-Shelf (COTS) components with the CMOS Omnivision CameraCubeChip^{TM} as its camera chip. The camera chip is protected by a sapphire window and takes pictures of 640x480 pixels in RGB. The structure of all the components is shown in the exploded view in Figure 4.2. The frame rate is currently one frame per three seconds, and the Field of View (FOV) is 65 degrees. SHRIMP is designed to survive launch, transit, and the landing stage of the mission. However, it is still uncertain if the image quality will be ensured when operating outside the temperature range provided by the manufacturer (-30°C ~ 70°C). The temperature can rise up to 197 °C on the Moon's surface [41]. Power computation of SHRIMP is very low while it only consumes one watt at maximum during operation. Besides the integrated camera chip, a processor, voltage regulators, FRAM and an IMU, which includes six-axis accelerometers, are also integrated on the chip. Software for operating the chip is developed by the Lunar Zebro team, so TRON can communicate with the SHRIMP via the RS485 bus. Within TRON, a SHRIMP app is developed to fulfil the modular design approach of TRON. The final SHRIMP module is presented in Figure 4.1.

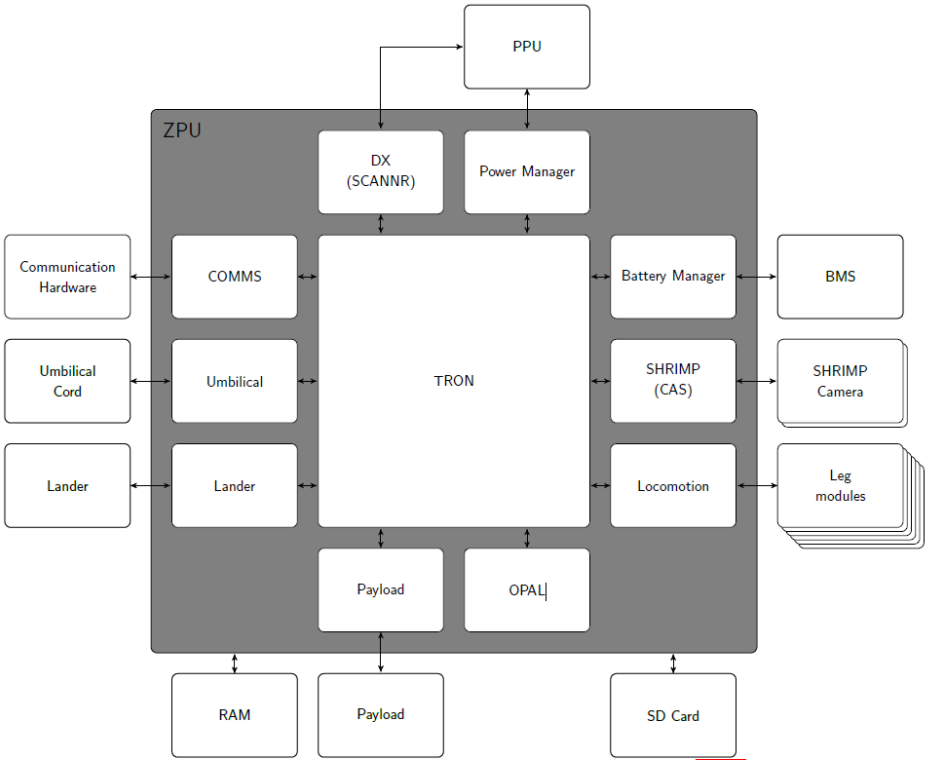


Figure 4.3: The Architecture of the ZPU of the Lunar Zebro [25]

4.1.3. TRON

TRON is the umbrella name for the controlling software on the OBC. This software is developed by Nick van Enthoven [25]. Most processes are run in sub-processes, also called apps. The communication between these apps goes through socket connections. TRON controls these sub-processes by restarting apps when they stop working, are timed out, or when they output an unexpected result. The architecture of the Lunar Zebro on software level is depicted in Figure 4.3. In this figure, the modularity of the system is clearly visible.

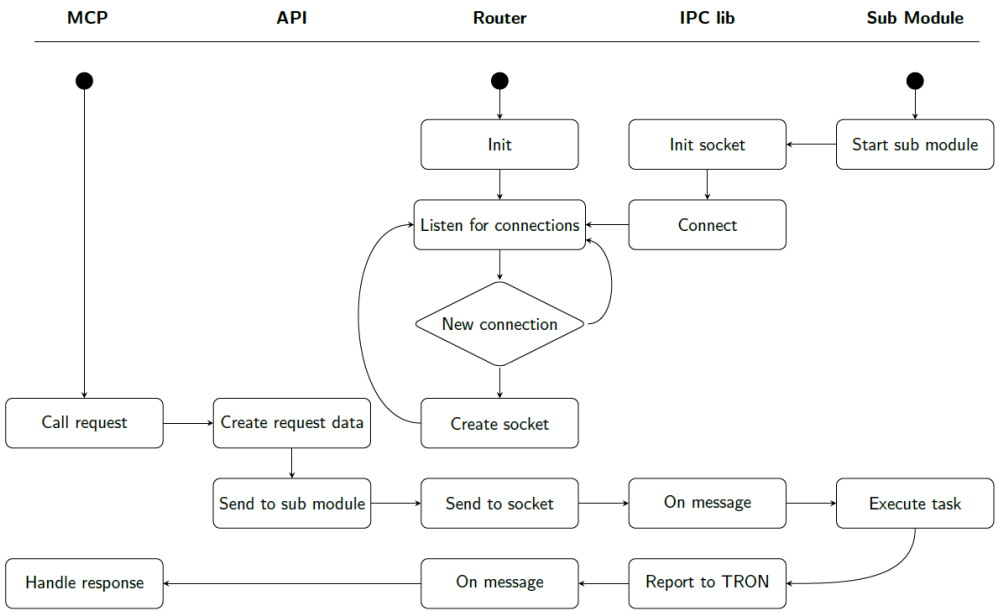


Figure 4.4: The Activity Flow between all the TRON components [25]

The inter-process communication, as described, is via direct TCP/IP, using POSIX sockets. A router is design to manage the socket connections and to relay messages. A library is written to ease the use of this IPC router. The activity flow of all the processes is shown in Figure 4.4. In this figure, all the activities between the MCP, the API of the MCP, the router, the library of the router, and the sub-process are elaborated on.

During the development of TRON, van Enthoven focused on meeting requirements to make the software space graded. When deployed on the moon, it will be hard to fix a crash or a deadlock of the software. It is, however, critical for the mission that these software modules will continue operating.

4.2. Functional Analysis

The purpose of the functional analysis is to determine the main and sub-functions of the system. The functions are connected to performance requirements in a later phase. For OPAL, the main functions are detecting hazards of the surface on the moon. It requires, therefore, some sub-functions to be executed in order to see these hazards. These sub-functions are based on the computer vision theory stated in section 2.3. Initially, for a computer vision system, calibration is required to make the robot aware of the camera's position relative to its surrounding. Next, the SHRIMP camera's input can be pre-processed before translating these pictures into a 3D point cloud. At this point, the obstacle must be identified and quantified. All the algorithms need to be executed properly, and errors in these executions need to be caught. Finally, the output and its execution status need to be communicated to TRON. All these functions working together will help OPAL detect obstacles. These functions could form a tree with their own sub-functions. This tree is illustrated in Figure 4.5.

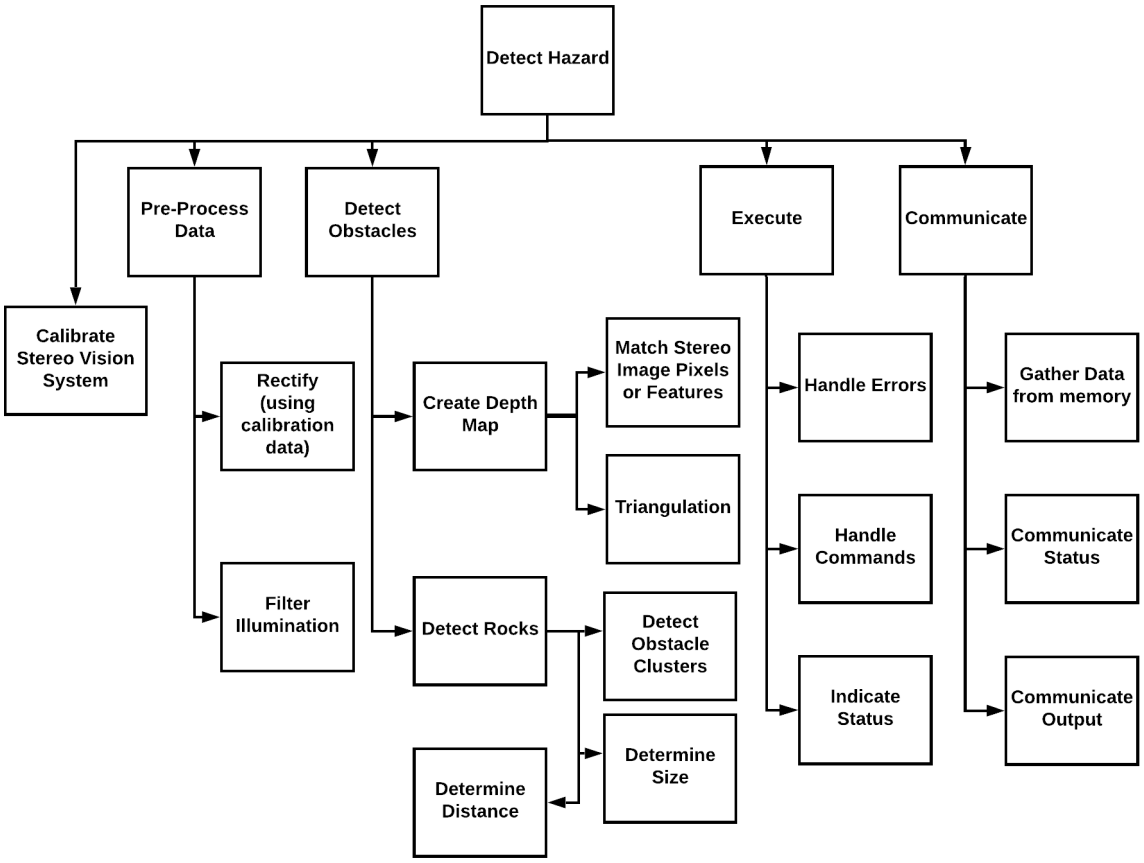


Figure 4.5: Functional Breakdown Tree

4.2.1. Key Requirements

Some of the key requirements that follow from the functional analysis are further substantiated in this section. A rationale is given for the requirements theme selves and, most importantly, to the value attached to these requirements.

One of the most important requirements is the height of the positive object in order for the obstacle-detection system to be able to detect it. This height appeared to be 30 mm while testing at the Mars-yard of Decos (for

more information see appendix B). The rover struggled over rocks larger than this 30 mm height. The problem was that the rover could become planted itself on its 'belly' on a rock while not being able to touch the ground any longer. Climbable obstacles are quantified to be at least smaller than the height between the bottom of the leg and the body's bottom when standing, so as to prevent becoming stuck. The grooves on the legs are not to be taken into account. This height is presented in Figure 4.6.

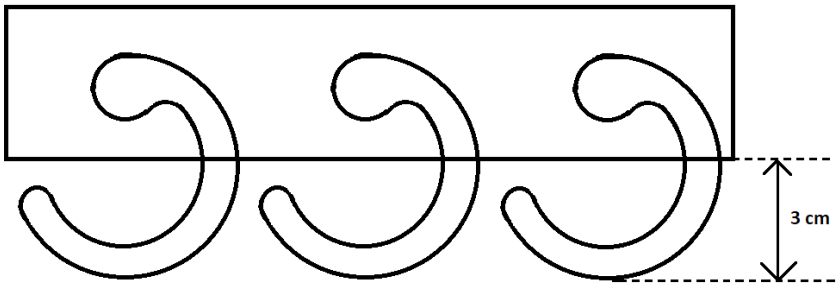


Figure 4.6: Indicating the height for the bottom of leg to the bottom of the zebro

Another key requirement is the closest allowable distance to a rock. This distance is determined using trigonometry and logic rules. The Lunar Zebro is only able to turn on the spot or walk straight. If a rock is within the turning circle, as visualised in Figure 4.7, the Zebro would crash. The field of view and straight tangent lines in walking direction, defines a line. After this line a rock could be in the so called blind-spot of the camera system. The Lunar Zebro could, in a worst case, make a step of 14 cm. This happens when the rover is laying on its belly, and the motors make one full revolution while having a 100% grip on both sides. So, when a rock is closer to the danger line than 14 cm a rock could be in this blind-spot if this worst case occur. The theoretical depth error at the distance of 48.6 cm is around 4.8 cm. This theoretical error is calculated in Section 7.1. So if the measured distance to the rock is closer than 53.4 cm (48.6 cm + 4.8 cm) the rover has to stop and start manoeuvring.

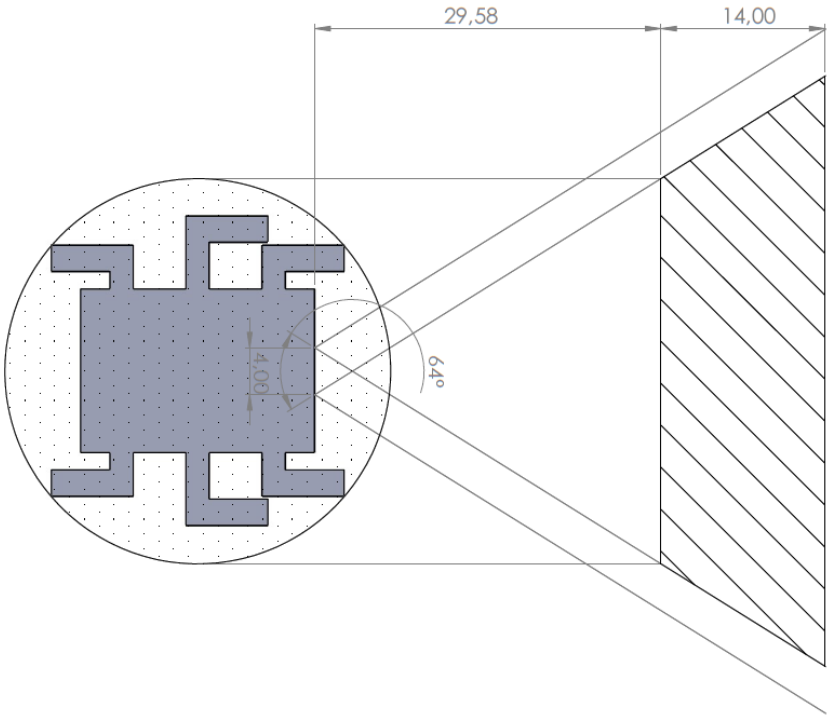


Figure 4.7: Indicating of the minimum distance to an obstacle + dashed 'Danger Zone'

The mission objectives define the time reserved for obstacle processing. One of the main goals of Lunar Zebro

is to walk for 200 m in one lunar day, which equals 14 earth days. Of those 14 days, the team estimated that the Lunar Zebro could use 10 days for walking. The Lunar Zebro could operate around the six hours on one battery charge with a capacity of 12800 mAh. Where the charging time with this amount of power is around five hours and 15 minutes. This means that in 24 hours, the Lunar Zebro can perform two walking cycles. Two cycles per 24 hours means that in every cycle 10 m of ground need to be covered. With the walking speed of the Lunar Zebro, it takes around 100 minutes to finish this 10 meter distance. It is estimated a photo is taken every 5cm, which is equal to the error at the detection range, therefore, OPAL will produce 200 photos per cycles. The time it takes to take a stereo photo and send it over the communication bus is a maximum of 6 seconds. The available time left for OPAL processing one stereo pair is around 73 seconds.

4.3. Requirements List

Both the interfaces and the expected functionalities are translated into requirements. Both TRON and SHRIMP define OPAL's interface requirements. For proper implementation of OPAL on the Lunar Zebro these prerequisites need to be met. All the interface requirements are listed in Table 4.2. The expected functionalities are translated into performance requirements. The performance requirements are listed in Table 4.3. The execution and communication functions were already defined by the interface requirements and have been omitted in this list. Both tables also provide a rationale for each requirement as well as a verification method. These methods were discussed in Section 3.5. The verification methods of the first couple interface requirements are left red, since they cannot be validated in the thesis because of resource restrictions.

Table 4.2: The Interface Requirements of OPAL

ID	Requirement	Rationale	Validation Method
OPAL-INT-01	OPAL shall be run using the IPC library of TRON	The Inter-Process Communication (IPC) library for communication between the Master Control Program named TRON and its sub-modules. This library will ask TRON to set up a socket connection between the sub-module and TRON and forward and receive messages.	
OPAL-INT-02	OPAL shall report its output, a distance to the closest obstacle, and error/status codes to TRON using the IPC library of TRON.	See N2-Chart.	
OPAL-INT-03	OPAL shall report errors and statuses using codes defined by TRON.	Predefined error and status codes helps the team to easily identify problems when in a more developed stage.	
OPAL-INT-04	OPAL shall have as input the images produced by two SHRIMP cameras with a stereo base of 40 mm, a resolution of 640 x 480 pixels and a field of view of 65 degrees.	This is the current implemented stereo vision system on the Engineering Model of the Lunar Zebro.	I
OPAL-INT-05	OPAL shall be able to be run on the ZPU, which has 500 MHz processor and is Linux based.	The ZPU is currently the CP400.8 of Hyperion Technology.	D
OPAL-INT-06	OPAL shall not use more than the 400 MB of RAM.	The RAM capacity of the CP400.8 of Hyperion Technology minus some RAM for other sub-processes.	D

Table 4.3: The Performance Requirements of OPAL

ID	Requirement	Rational	Validation Method
OPAL-FUN-01	OPAL shall be calibrated with a re-projection error lower than 0.2 pixels.	Performing multiple calibrations with Matlab (see Appendix C) showed that calibrations with a re-projection error higher than 0.2 showed that there were unclear pictures, and the results could not be used for rectification and stereo matching	D
OPAL-FUN-02	OPAL shall limit the contrast with a maximum change in intensity to limit the influence of lighting on the algorithm.	Sun will have a negative influence of the performance of an obstacle-detection algorithm, because it will change the lighting overtime. OPAL shall try to minimize this influence.	D
OPAL-FUN-03	OPAL shall detect the presence of rocks larger than 30 mm in front of the Lunar Zebro.	Rocks larger than 30 cm are a risk (See Section 4.2.1).	T
OPAL-FUN-04	OPAL shall detect hazards within the detection area as pointed out by Figure 4.7.	In this slot, the obstacles are becoming hazards to the Zebro (See section 4.2.1).	T
OPAL-FUN-05	OPAL shall determine the distance to every hazard with an error less than the theoretical error curvature.	This worst-case theoretical error is explained in section 2.3.7.	T
OPAL-FUN-06	OPAL shall not exceed the time limit of 73 seconds to process a frame.	The time limit is framed by the mission objectives and the day-to-day activities of the Zebro (See section 4.2.1)	T

5

Design of Bars, the Test Model

For the development of the stereo vision obstacle-detection system, collecting data is crucial. Since there was no stereo vision data within the team, the need for a collection platform or test model was high. Furthermore, test opportunities arose at the beginning of 2020. For example, the EuroMoonMars (EMM) team was planning a mission to Iceland and Italy for some analogue field testing, where they planned to simulate live science conducting on the lunar surface [87]. At these analogue testing fields, the Lunar Zebro could walk on some rough, close-to-reality terrain while collecting interesting data. With this in mind, the option to redesign the Lunar Zebro for these test missions became more and more attractive. Hence, the development of this new test model was started. Due to the Covid-19 pandemic, unfortunately, these missions were postponed and eventually cancelled. The development of the test model was already in an advanced stage and was therefore continued.

In this chapter, the development of this new test model is reported. At first, previous test models are discussed and, after this, the goals of the new test model are explained. Design and adaption choices are noted next, and then the final result is discussed.

Figure 5.1: DeciZebro with external stereo camera

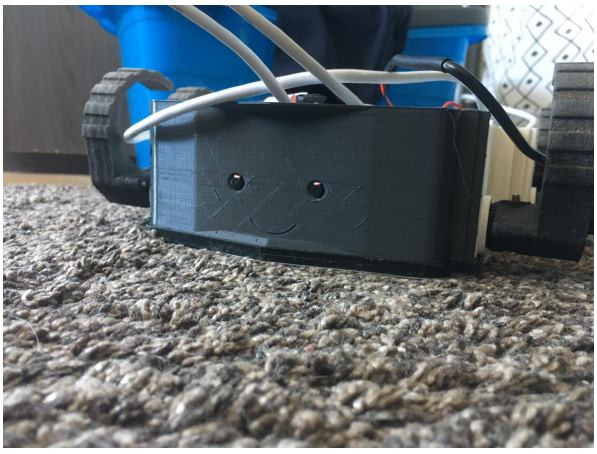


Figure 5.2: DeciZebro equipped with a StereoPi and two RPi Cameras

5.1. DeciZebro Test Models

At first the DeciZebro seemed to be the perfect candidate to function as a testing platform, since it was in an already advanced development stage and only required some small adaptions. Multiple attempts were made to adapt the DeciZebro into a trustworthy working platform. On the first try, a stereo camera owned by the team was connected to the DeciZebro. Unfortunately, this camera could not be controlled by the Raspberry Pi since it required a 100% CPU-load. While using this high CPU demand, the Zebro processed only 0.3 frames

per second. This would mean that the Zebro could not record while walking. Connecting the camera to another computer while walking was a solution that worked; but this required a long usb-cable which suppresses the flexibility of the system. This solution, nevertheless, produced some initial footage. The stereo camera and the mount are shown in Figure 5.1.

A Stereo Pi [95] was introduced to enable on-board parallel recording and walking. This Pi is replacing the Raspberry Pi 4. In section 5.3.1, the choice of this option is substantiated. With the new Pi, a new front was designed and printed, and new cameras were added. The result of all this work is shown in Figure 5.2. However, due to the aggressive walking behaviour and the completely different master control program (ROS), the DeciZebro was eventually considered unsuitable for the opportunities discussed in the introduction. It was decided that a completely new test model of the Lunar Zebro would be designed.

5.2. Test Model Goals

In 2017, the electronic engineers of the Lunar Zebro started working on a functional electronic design. The focus lay entirely on the hardware, seeking to make sure it would fulfil all the technical requirements. When the design was finished, the team produced and assembled their first model. Besides an excellent electronic design, the flexibility and ease of assembly were two elements that were missing. This problem was already experienced in previous designs of the terrestrial Zebro. Otten's DeciZebro [81] had taken into account the Design for Assembly (DfA) principle described by Boothroyd & Alting [8]. When scaling up in a swarm, it is better to have a quickly assembled sequence as this would also reduce the risk of breaking things when disassembling and re-assembling. This principle revealed itself when the current design of the motherboard (Figure 5.4) broke after some assembly cycles. Connectors became loose, and chips were not protected with Electro Static Discharge (ESD) protection. There was also not much room inside the Zebro, which caused problems when including external on-board computers, batteries, and DC/DC converters.

Therefore, the goal has been to develop a reliable, robust, and easily assembled test model that could host vital flight hardware like the motor drivers, the BMS, and the SHRIMP cameras; this test model will be a platform for testing and optimising all these systems in practice. To achieve this, a new carrier board needs to be designed and manufactured, and the body and legs need to be adapted. Furthermore, the test model needs to be equipped with software similar to the expected flight software. Thus, the overall end goal is a controllable walking test Lunar Zebro called Bars.[1]

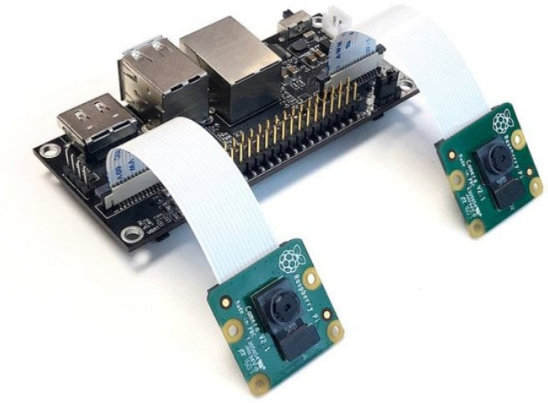


Figure 5.3: A Stereo Pi[95]

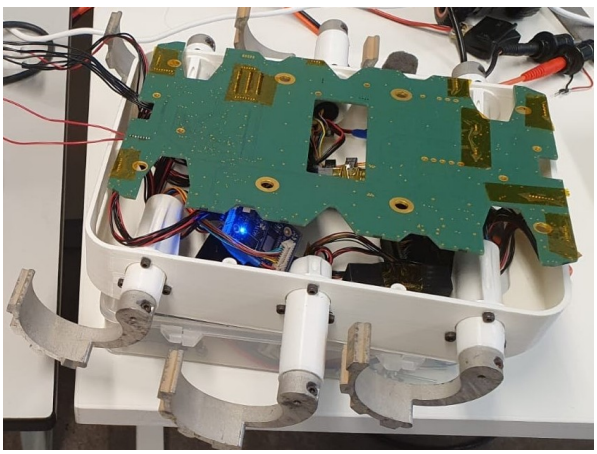


Figure 5.4: The old motherboard of the lunar zebro

[1] All models of the Lunar Zebro are named after Russian space dogs (Belka, Strelka, and Leika). Bars was their pioneer. Bars died together with another dog in the launch before Belka and Strelka were launched. The Russian translation of Bars is Snow Leopard, which fits the body that was finally developed.

5.3. Design Choices and Adaptions

In this section the design choices that were made during the development of Bars are clarified. The on-board computer has been upgraded to a Stereo Pi [95], and the reason for this is elaborated on in Section 5.3.1. In Section 5.3.2, the camera design is highlighted. Extensive improvements have been made to the PCB as well, and the most important design decisions are presented in Section 5.3.3. Furthermore, the body and legs of the Zebro have been slightly altered, as described in Section 5.3.4. Finally, in Section 5.3.5, the changes in the software design are discussed.

5.3.1. On-Board Computer

There are a couple of reasons why the CP400.85 on-board computer of Hyperion [68] is not used in the development of the test model. Since this will be flight hardware, this would have been the most convenient option. However, the team lacks experience with this board. The reason for this is that there is no development board to test some crucial software and hardware interfaces. Each interface requires development, which makes it less attractive for fast prototyping. Interface flexibility is attractive, especially when using Commercial Off-The-Shelf (COTS) products.

Among the available computing platform alternatives, Raspberry Pi was selected as the replacement on-board computer for the development phase. The Raspberry Pi foundation has developed single board computers since 2012 [33]. They produce inexpensive boards based on ARM processors, and there are already multiple versions on the market. Most Raspberry Pis are equipped with a Camera Serial Interface (CSI) and use a small GPU to process the camera footage, making sure that the CPU is not overloaded. These features make it perfect for on-board recording and processing. However, there is only one CSI port on a standard Raspberry Pi. The Raspberry Pi Foundation has also developed a Compute Module with all the breakouts, like a standard Raspberry PI, but broken out to a DDR2 SO-DIMM connector. Besides all the standard connections, there is a second CSI interface. A company called virt2real [95] made a board that transferred this DDR2 SO-DIMM back to all the standard Raspberry Pi board connections but added a second CSI camera interface. Both the Raspberry Pi Compute Module 3 and the CP400.85 use ARMv7 processors and use a Linux-based operating system. Thus, the Stereo Pi, which is what the board of virt2real is called, is perfect for the new test model since it is easy to implement and close to the flight-ware.

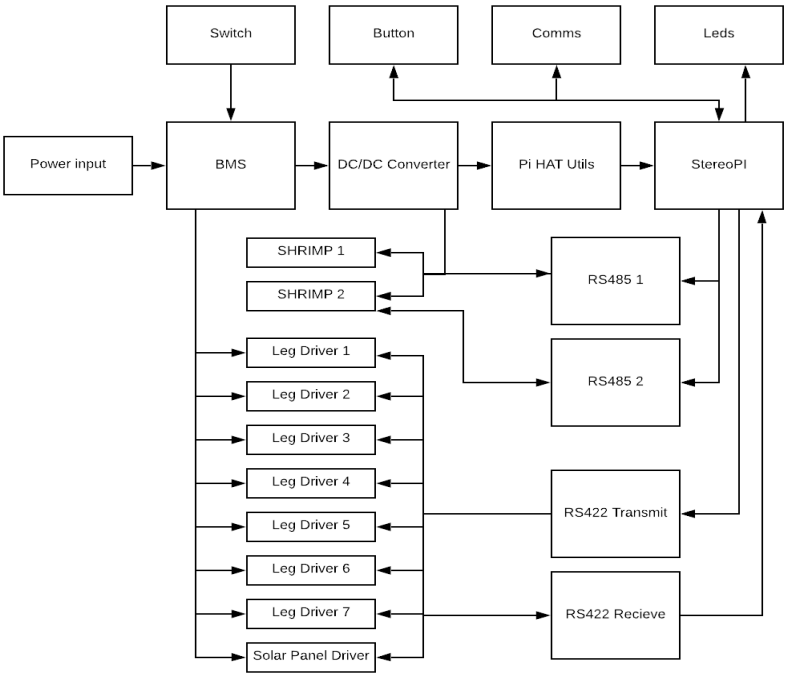


Figure 5.5: Architecture of the new carrier PCB of the Test Model

5.3.2. Camera

For the camera, a PiCam is used. This PiCam is specially developed for the Raspberry Pi. It is one of the least expensive cameras on the market with the required CSI interface, and it is already taking pictures with more megapixels than the SHRIMP camera. The PiCam has a 5 megapixels OmniVision OV5647 camera module, whereas the SHRIMP camera only has a 0.3 megapixels OmniVision OV7690 camera module [1]. The PiCam output is, however, downscaled to match the output size of the SHRIMP pictures.

The full camera interfacing setup is shown in Figure 5.3, where the Stereo Pi board computer is shown. This picture clearly shows how both cameras are connected via the CSI interfaces.

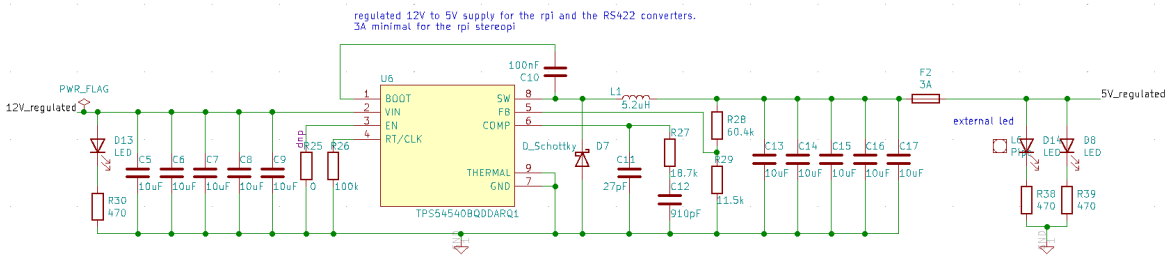


Figure 5.6: DC/DC conversion on the new carrier board

5.3.3. PCB design

A new PCB is designed in KiCAD, which is a PCB-designing software. As stated before, the focus of the design was on the ease of assembly. Clearly, in Figure 5.4, it is visible that the old motherboard is blocking access inwards. Inside the rovers it seems chaotic because the cables and motor drivers are not structured. A new structure should tackle both problems. This new structure or architecture is shown in Figure 5.5. It can be seen that the motherboard's primary function is providing power and communication to all its components.

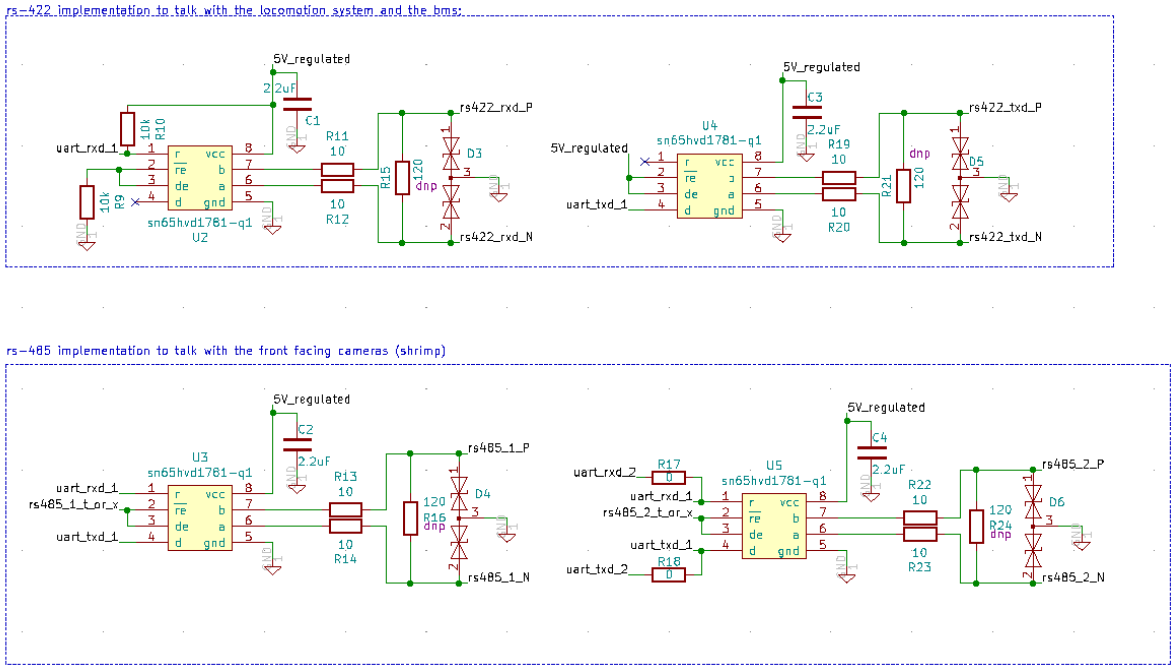


Figure 5.7: RS485/RS422 implementation on the new carrier board

The power comes from the BMS which will act as a 12V regulated power supply. This voltage is directly distributed to the motor drivers. However, the Stereo Pi requires a 5V supply voltage. So, an on-board non-isolated DC/DC conversion is used to reduce this voltage. The circuit performing this conversion is shown in Figure 5.6. The DC/DC conversion chip is chosen, using the characteristics of a Raspberry Pi, which requires a power

supply of 5V and 2.5A. But the Stereo Pi is not the only element using the 5V power source. The additional 5A is intended for powering the rest of the components. The power circuit is drawn using the spec-sheet of the power chip. The carrier board also has some protection if the Raspberry Pi is powered externally, based on examples integrated into KiCAD.

Communication between the motor drivers and on-board computer passes through an RS422 bus, where the camera uses a RS485 bus. The benefits of these communication protocols are that they are very reliable, due to the use of differential pairs which makes it less sensitive to noise and electromagnetic interference. The RS422 bus is a master–slave configuration which supports multiple devices, whereas the RS485 is a point-to-point connection. The data rate of both busses is up to 10 MB per second. On the PCB, these communication busses are implemented as shown in Figure 5.7. The circuit used is the same as that used on the motor-driver developed by Jeffrey Miog [67]. All the busses are connected and controlled by the UART busses on the Stereo Pi.

Besides power and communication, the carrier board was given additional tasks. The motor driver connectors have been changed to stacking connectors to ensure they are placed closer to motors and sensors, better organising the cabling inside the rover. For the Stereo Pi, there is a stacking connector as well. Moreover, the new carrier board hosts a 433MHz receiver, some switches, button, and some LEDs to ease operations and support remote control. During development, 3D modelling was used to ensure the components' configuration is fitting into Bars. The 3D model is shown in Figure 5.9. The final product without all the connectors is depicted in Figure 5.8.

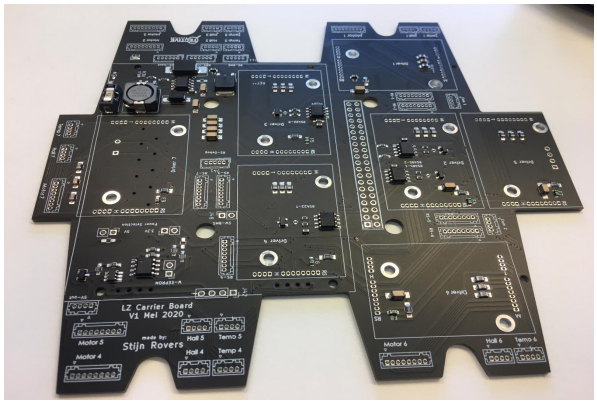


Figure 5.8: The new carrier board without connectors

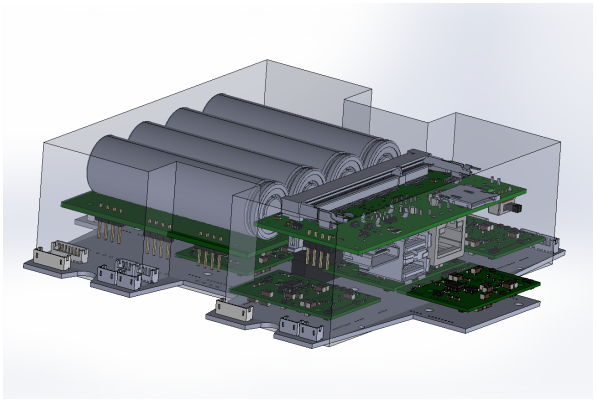


Figure 5.9: The 3d model of the new carrier board

5.3.4. Body and Leg design

The body of Bars is 3D printed; this was done to reduce cost and to speed up the prototyping process, making it easier to adapt for project changes. The final design was not ready for 3D printing, but it required some adaptations to make it printable. These improvements increased the printing process speed by 300%. Besides speeding up the printing process, the adaptions increased the quality. For example, the Frangibolt release system's structural reinforcements, which will not be tested with Bars, could easily be removed. Another adaption is separating the 'head' of Bars from its body. The head is shown in Figure 5.10, with the rest of the body in Figure 5.11. This separation will reduce the amount of needed supporting material. After the first iteration, it turned out that the thickness that causes the body's strength was not sufficient. Thereafter, the thickness of the body was increased by 1 mm to 3 mm.

Besides the body, a new holder for the Raspberry Pi cameras was needed to meet the SHRIMP camera's interface requirements. After measuring both mounting interfaces, the camera holder was designed in such a way that it was printable. The holder went through several iterations because a 3D printer requires weak tolerances. The end result is shown in Figure 5.12.

In the end, the leg design faced some structural problems. The interface with the motor shaft was not sufficient, and the leg came loose while walking. After printing multiple legs with different tolerances using single or multiple screws on the side for clamping, a simple extra bolt clamping perpendicular on the shaft turned

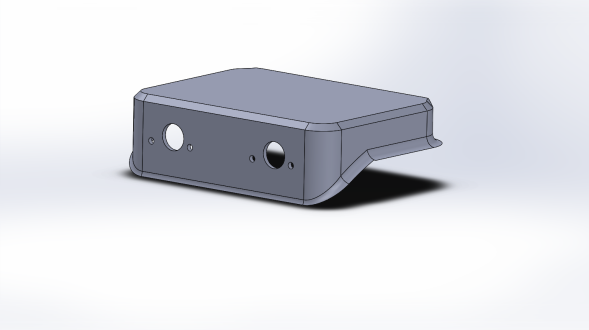


Figure 5.10: "Head" of the test model

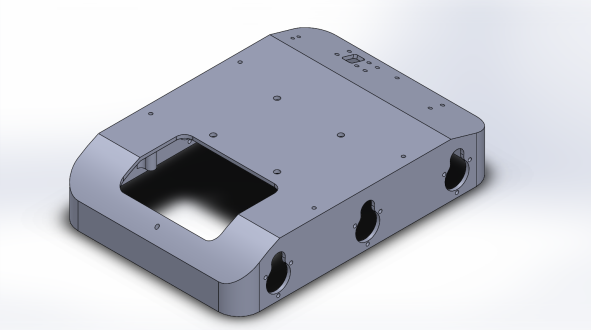


Figure 5.11: Body of the test model

out to be the best solution. However, since the d-shaft was small and the torque was not neglectable, the shaft started to wring into the 3D printed material, which caused the leg to come loose again. Finally, the same solution was applied to the old aluminium legs, as shown in Figure 5.13. Later, a new interface will need to be designed to minimise the forces on the 3D plastic material. For example, a hub from the motor shaft to the leg could fix this. Thereafter, the legs could be produced from 3D printed material once again.

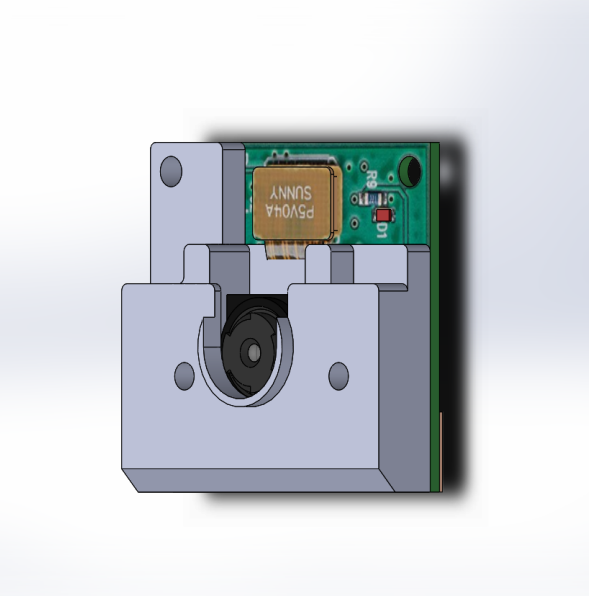


Figure 5.12: The Cam Holder

Figure 5.13: The Leg Fix

5.3.5. Software Integration

The main controlling program implemented on Bars is TRON. A new state for TRON is written primarily for the test mission. This state will control some added emulators. A SHRIMP emulator emulates the functionality of the SHRIMP camera while using the Raspberry Pi cameras. The emulator is able to start the Raspberry Pi camera; it can start and stop a recording, and it is able to take pictures. Furthermore, there is a communication module emulator, which translates the commands received from the 433MHz remote controller.

A new locomotion algorithm has been created to have a stable gait. The software developed by Rouwen contained unexpected behaviour since it was based on Max-Plus algebra [88]. The DeciZebro team looked for more reliable gait-switching methods and brought back the original RHex gait-switching algorithm of Saranli

et al. [90]. The DeciZebro has walked many metres with this algorithm and has shown promising results. Therefore, this algorithm is translated to C++ and adapted for operation on the Lunar Zebro motor driver's communication protocols.

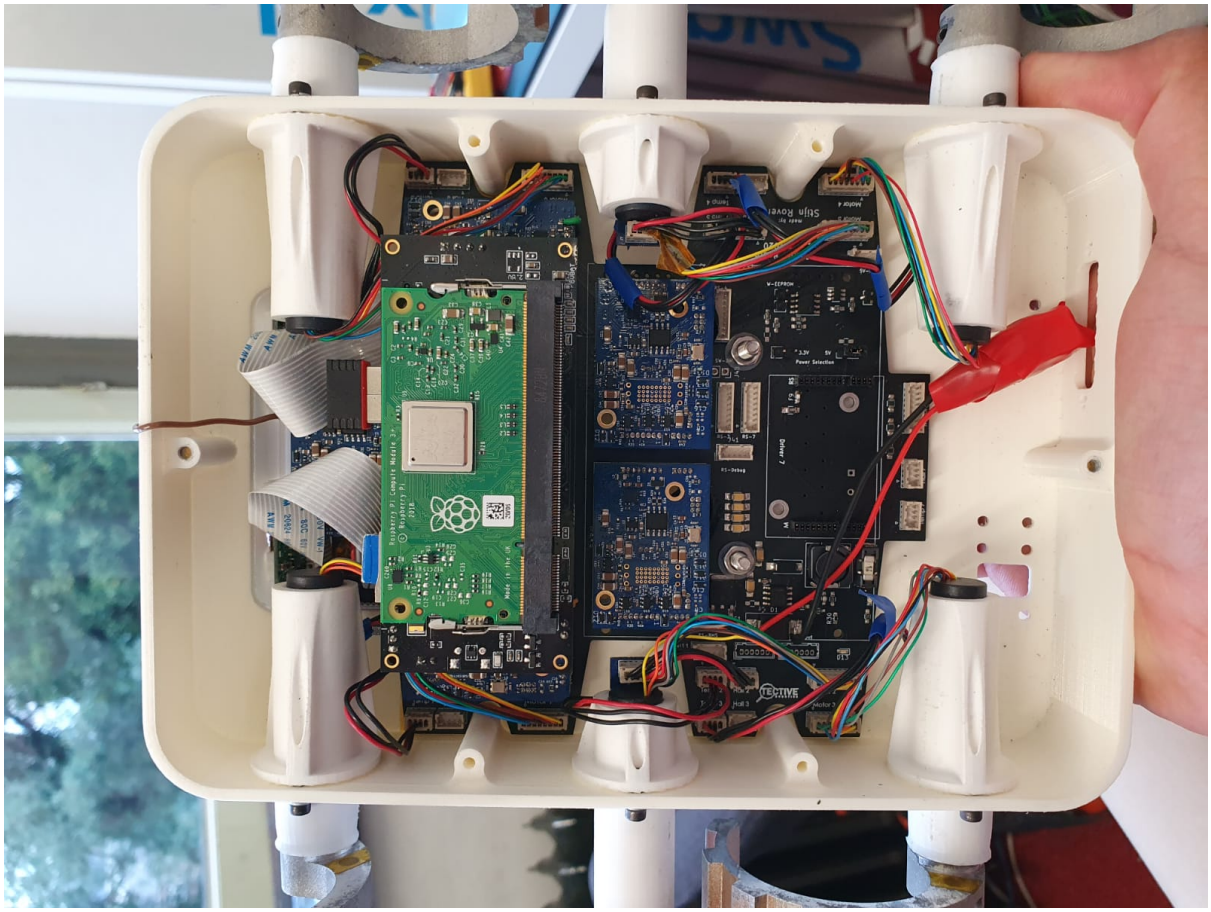


Figure 5.14: Look inside the new test model

5.4. Testing Bars

In order to use Bars as a test model for the subsystems, Bars needs testing first. The electronics were tested in small chunks during soldering. For example, the DC/DC converter has been tested first before connecting it to the Pi. The same was done with the communications receiver; a command was sent and decoded manually using an oscilloscope before the system could be integrated.

After the electronics were successfully tested, the legs were mounted, and its first steps were made. The first demonstration was on a plastic box with Bars' legs spinning in the air; later some steps were set on the ground after some parameter and code refinements. Before the stereo vision test at the Decos test facility, the locomotion was first given a field test at the same facility to make sure every was working on site. The tests which were conducted are listed in Table 6.1. The tests contained real basic operations, like walking in a forward motion, as well as more complicated tests like climbing obstacles. Ahmed Abakay, who helped rewrite the locomotion software, mainly operated these tests.
The field tests showed out that Bars was able to walk in a forward direction at all the different speeds as well as turn on the spot. The rover's body was hitting the ground during one of its first steps only at 100% walking speed. Minimal deviations were detected when forward walking on a flat surface. When walking instead on small rocks, these deviations became greater. On a slope, the rover was performing sufficiently. The rover could walk up to slopes of 25 degrees. When walking obstacles of 30 mm or higher, the locomotion started having problems and could get stuck on top of the rock. The other problem when facing the rock obstacles was that the leg produced an over-current when it got stuck and stopped working. During operation, one out

Table 5.1: Locomotion Test List

Nr.	Test Description
1	10-100 % Forward
2	10-100 % Left
3	10-100 % Right
4	Multiple inclination angle measured with the phone
5	Multiple 'walkable' obstacles

of 10 times, one or multiple legs stopped working for some reason. The reason for this is still unknown, but this problem's origin will probably be in the safety protocol of the motor drivers.

The other new functions, such as the camera system and the remote control, were also functioning as planned. During the walking, TRON could record the camera feed and still be controlled using the remote.

Figure 5.15: The new test model assembled

5.5. Result

After several months of work, a fully working test Lunar Zebro called Bars has been realised. Bars performs test missions and behaves like the final Lunar Zebro model. The test model is shown in Figure 5.15, standing on a rock at the Decos Mars yard. The inside is visible in Figure 5.14, containing the newly developed carrier-board with all the components stacked. Because cables are nicely organised on the side of the PCB, the internal space has increased considerably compared to the previous model. Bars is performing sufficiently for gathering usable stereo vision data from the rover perspective for the development of OPAL.

6

Obstacle Detection Algorithm Design

In this chapter, an obstacle detection algorithm is designed. As Matthies, a computer scientist who worked for NASA, defined it: 'Obstacle detection is the process of using sensors, data structures, and algorithms to detect objects or terrain types that impede motion.' [63]. From the requirements in section 4.2, the sensors are the stereo cameras of the Lunar Zebro and the obstacles that impede motion showed mainly as rocks that are greater than 3 cm. Section 2.3 elaborated on the basic computer vision theory needed to substantiate the obstacle detection system's design process. The main goal of this design chapter is to elaborate on the chosen algorithms and design choices for Obstacle Processing ALgorithm (OPAL). First, Section 6.1 provides an algorithm overview. Following this, every discussed block in this overview is clarified.

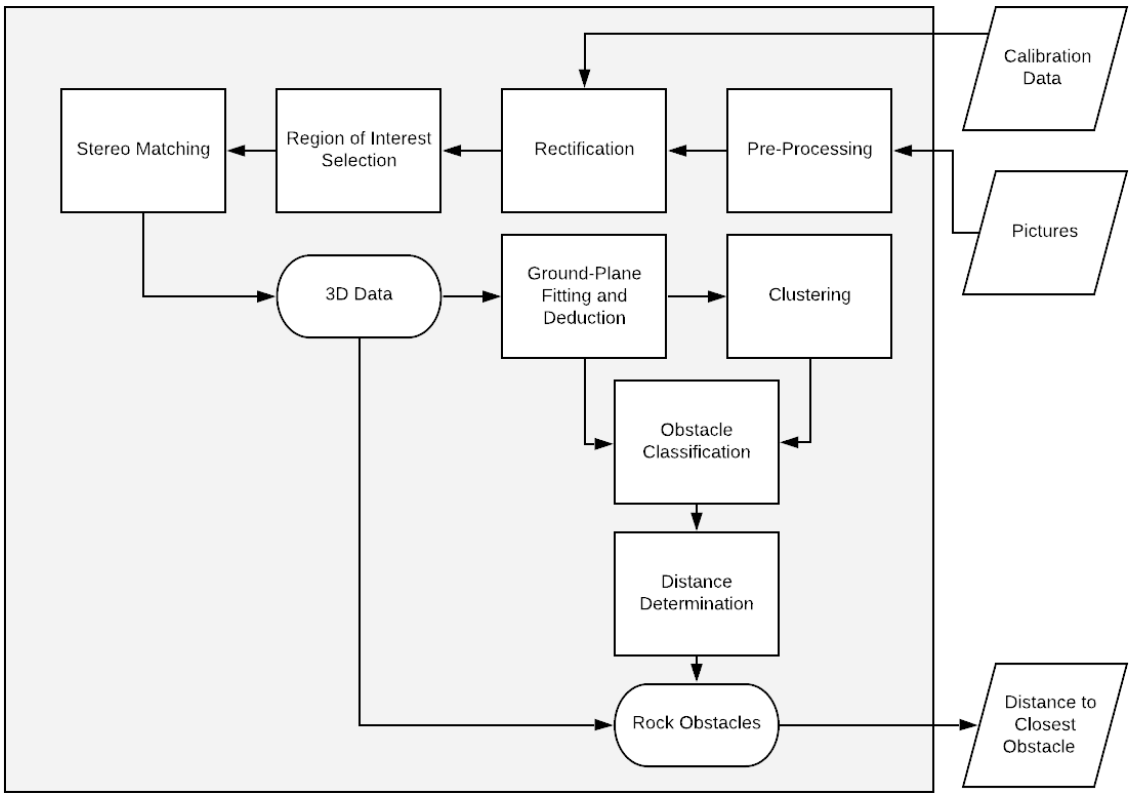


Figure 6.1: The Architecture of OPAL

6.1. System Overview

Section 2.3 discusses some basic steps that could help a computer determine objects or obstacles in the scenery. At first, a camera model needs to be gathered, which is done in this calibration step. After that, rectification using the camera model can speed up the stereo matching processing, making sure depth can be measured. A structure or a strategy is set out to create a flow of these steps and algorithms. This flow will guide the system to subtract obstacles from the scene. The architecture or flow is shown in Figure 6.1 and is based on flowcharts used by Wang et al. [108], and Di et al. [22]. Both flow charts are used in rock detection algorithms for planetary navigation.

Stereo camera pictures are taken as input, together with camera parameters, to perform a first calibration, as discussed in the rectification section 6.3. The first block is a pre-processing step containing, for example, an illumination filter that filters the harsh illumination condition on the Moon's surface. Second, a region of interest selection is continued so as to remove irrelevant parts of the image such as the sky. In the end, this step will reduce the computational cost. Furthermore, the disparity values are filtered by fitting and deducting the ground plane. Moreover, the resulting rock segments and other segments are now clustered so that multiple rocks can be distinguished. Finally, logic is applied in obstacle classification to detect objects higher than 3 cm from the ground. The distances to the obstacles are determined in this last block. According to the requirements in section 4.2, the only output of OPAL will be the distance to the closest obstacle. At this point, the closest obstacle is communicated to TRON. All the blocks of the architecture are individually elaborated on in the upcoming sections in sequential order.

The majority of the algorithms used in this thesis is implemented in the Open Source Computer Vision (OpenCV) library. As the name explains the library is mainly for image and video processing. Twenty years ago the source code was released by Intel®. Eventually, more programmers contributed to further develop the library. The library already includes over 2500 optimised algorithms and is still growing. This work is paying off, because the library itself is downloaded over 2.5 million times, and is extensively used. The main advantage is that the library is cross platform; not only it can be used on multiple operating systems, but it can also be used with different program languages [10].

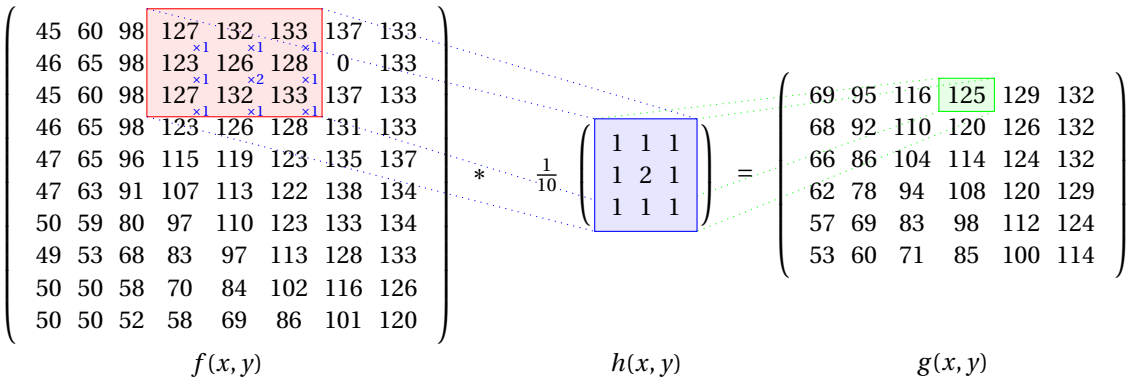


Figure 6.2: Convolution/filtering example [102]

6.2. Pre-Processing

Before any processing method is applied, the pictures are pre-processed with two image filters. Noise is a very common phenomenon while processing signals, and is the term used for unknown perturbations of a signal. A commonly used process to remove noise from signals is filtering. Typically for an image, one pixel is likely to be related to its neighbours. Therefore, most linear filters use a kernel, which are weighted as to how much the neighbouring pixels will influence the filtered result. In Figure 6.2, the convolution process using a kernel is visualised. This process can be summarised in the following formula:

$$g(i,j) = \sum_{k,l} f(i+k, j+l)h(k,l)$$

In which it can be noted that g is the result of the convolution of f and h, simply:

$$g = f * h$$

There are a couple of effects that filtering can achieve. A low-pass filter can smooth the image. It is a basic noise removal technique also used in signal processing. It removes all the high frequencies by averaging them out using kernel weights, also called weighted averaging. A high-pass filter sharpens a picture. It is the same technique but uses a different kernel. For example, a kernel containing the Gaussian derivative, typically used for defining the normal distribution curve, is sensitive to rapid changes in the pictures, such as edges.

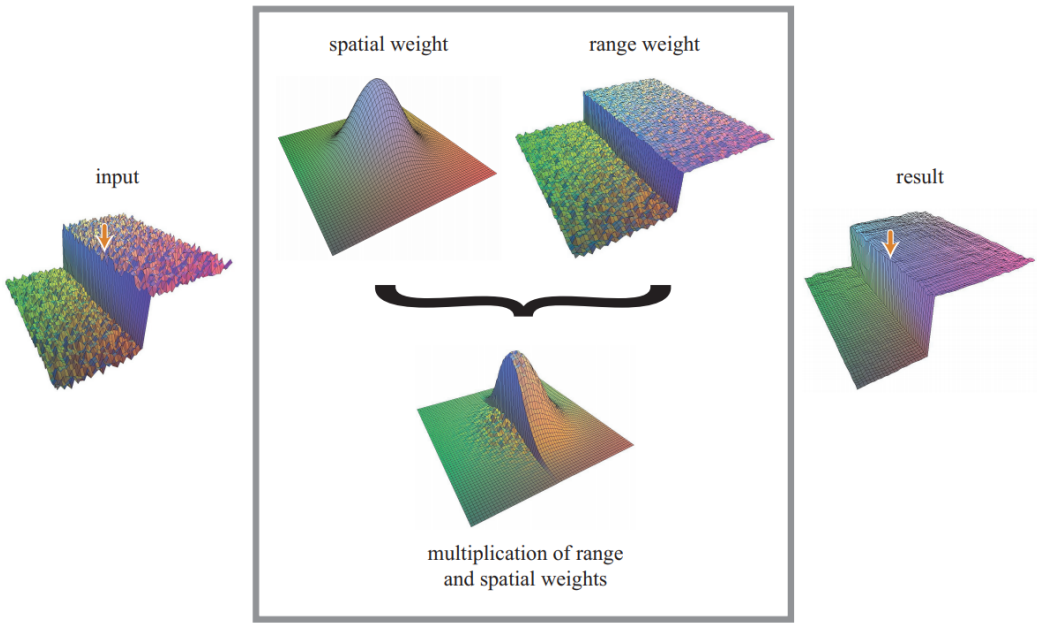


Figure 6.3: Bilateral Filtering Visualisation [101]

The bilateral filter showed that it could improve the stereo match quality [3]. This filter is an extension of a normal Gaussian filter, a weighted average that conserves the edges in the picture. How this is done is visualised in Figure 6.3.

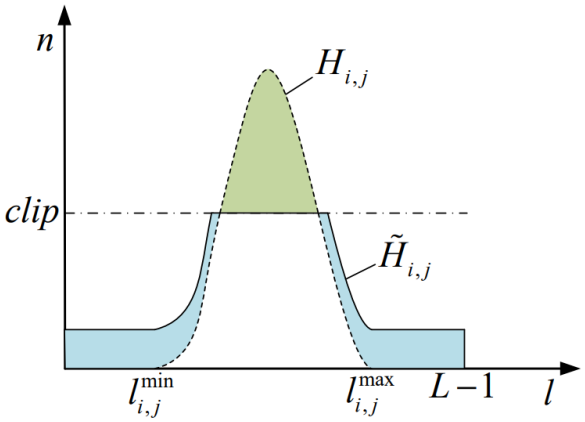


Figure 6.4: Histogram Clipping [57]

After the bilateral filter, an illumination filter is applied. This filter helps reduce the influence of low inclination angles of the sun, limiting long and harsh shadows, and helps also limiting lens flares or over-lightning. The

Contrast Limited Adaptive Histogram Equalization (CLAHE) method [109] [86] [12] is a widely used filter. This method uses an N x N square kernel around every pixel and constructs a histogram of this kernel. The peaks in the histograms are clipped and distributed over the kernel [104]. In Figure 6.4 this clipping process can be seen. In this way, it increases the local contrast without amplifying the noise. The CLAHE method is selected because it showed good performance with high illuminated scenes. It was originally designed to enhance performance while having foggy conditions.

The result of both filtering processes could be seen by comparing Figure 6.5 and 6.6. The first figure shows the original captured frame with some added lines, while the second figure shows the result after the filtering process. When looking at the cloud, for example, it can be seen that the contrast is enhanced in this overlighted region, and more cloud details could be spotted. The reason why the picture is turned into gray-scale to reduce the memory size of the picture and to make it possible to implement different stereo-matching algorithms.

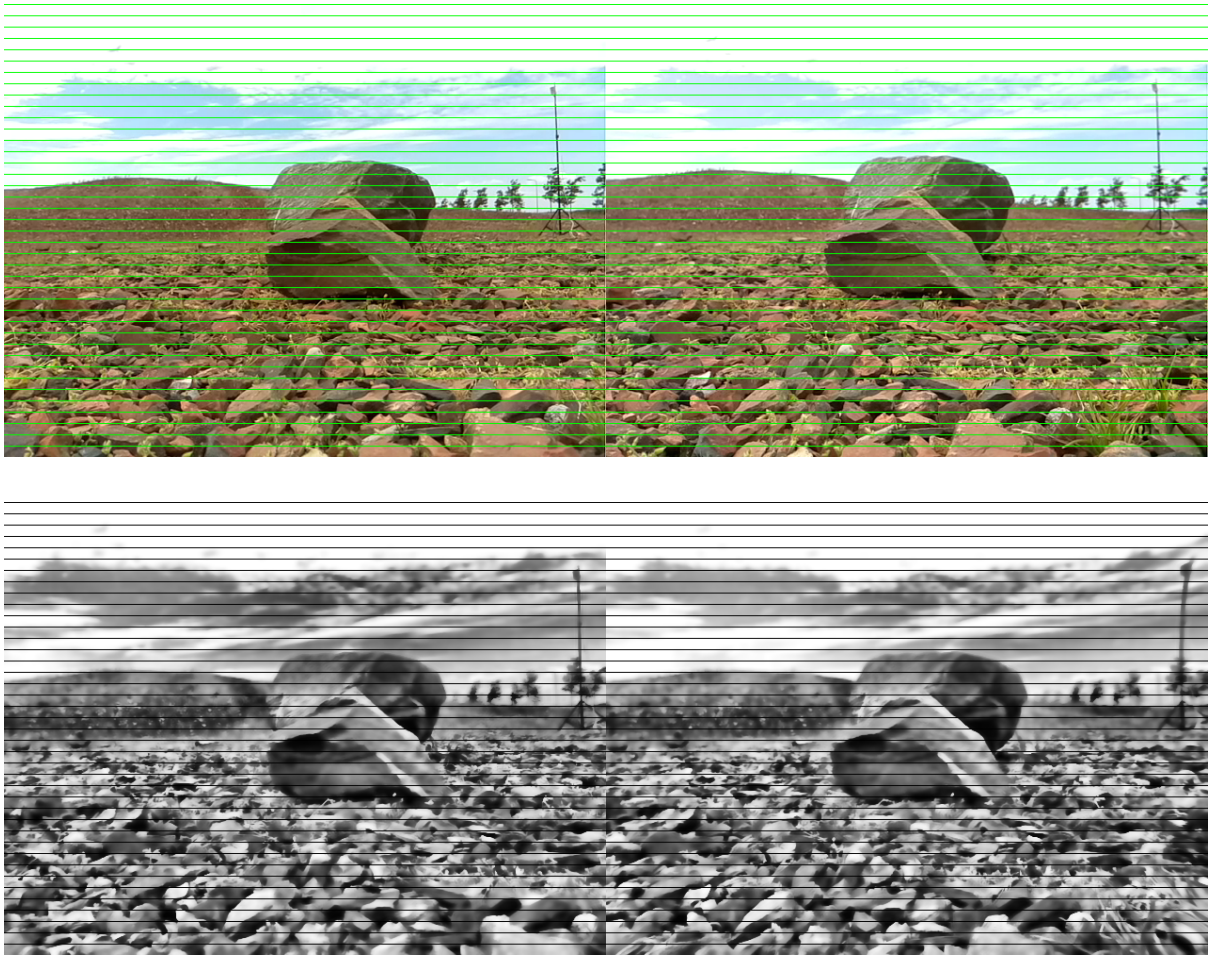


Figure 6.6: Frame rectified and filtered by the stereo camera with horizontal lines

6.3. Rectification

Rectification could help obtain 3D information of the scene. The main reason is to ease the stereo matching process. The rectification process will carry out two operations—distortion correction and the collinear alignment of both image plane's epipolar lines. The theory behind this process is described in Chapter 2. First, to perform rectification, the current stereo setup needs to be calibrated. The calibration process determines the intrinsic and extrinsic parameters of the stereo camera setup. This process involves making many pictures of a chequerboard or another calibration pattern in multiple positions. In Figure 6.7, the author gathers together

this calibration footage.

There are multiple tools around one can find the intrinsic and extrinsic parameters of the stereo camera setup. Popular tools are the Matlab calibration toolbox, and the algorithms implemented in the OpenCV library. Klimentjew et al. [49] tested both methods for obtaining calibration results, and the research showed that both methods demonstrated the same performance. For the reasons of ease and simplicity, the Matlab Toolbox has been selected. The calibration results are shown in Appendix C.

At this point, the pictures can be rectified using the stereo rectify function of the OpenCV library. The result can, besides the applied filters, also be seen when comparing in Figure 6.5 and Figure 6.6. The first figure is the original frame with horizontal lines added to visualise that the pixels in both pictures are not collinearly aligned. Where in the second picture, the image are rectified and the pixels in both picture does align. The visualised lines are a useful tool for seeing this phenomenon. For example, the line touches both the crater tops, while both ridges were not aligned in the original picture.

Figure 6.7: Performing Chequerboard Calibration

6.4. Region of Interest Selection

There are regions in the picture that contain barely any helpful information—for example, the sky. Therefore, it is not essential to use these in all the following computationally heavy steps. Hence, a Region Of Interest (ROI) selection is made to deduct these regions. Furthermore, some regions in the pictures, specifically at the sides, produces uncertain results. These uncertainties are the result of calibration and rectification steps. This process was not flawlessly performed at the sides for a reason—the lens aberrations. These side differences were discovered during development, where they caused some discontinuity in the disparity maps. In addition, the left camera has some objects in its field of view that are not in the field of view of the right camera.

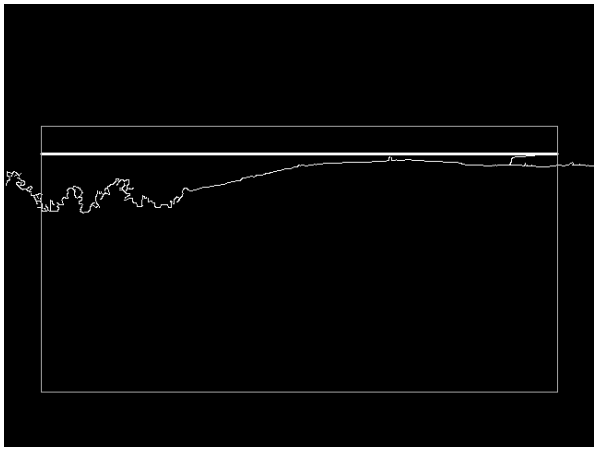


Figure 6.8: Canny Edge-Detection output and ROI selection lines

Figure 6.9: ROI output

This phenomenon is called an occlusion. Hence, both sides are deducted. The bottom only contains information of parts very close to the rover and therefore also seen as not important. The horizon is the only moving factor in the picture because the roll and pitch of the Lunar Zebro change while walking. So, in order to deduct the sky, this top boundary of the ROI is determined by an edge-detection algorithm. This algorithm tries to find the edge between the sky and the rest of the picture. At a later stage, the solar panel will be in the field of view as well. Fortunately, this algorithm can easily remove the lid as well.

The edge-detection algorithm used was initially developed by Canny [11]. The process of a Canny edge-detection algorithm is as follows. First, a Gaussian filter smoothed the image. Second, the intensity gradients in the image are determined. After this, a non-maximum suppression algorithm thins the edges. Following this, a hysteresis threshold method is used to determine the final edges in the picture. The Canny algorithm is outperforming the most commonly used edge-detection algorithms, according to Katiyar et al. [46].

In Figure 6.8, the Canny edge-detection output is shown. The bright horizontal line represents the first row where an edge is present. The picture represents the determined region of interest, which is 30 pixels higher than the detected top boundary. In Figure 6.9, the final output of the ROI Detection algorithm is shown. Next, all the black edges will be removed. At this point, a smaller picture is pushed to the next blocks.

A small performance test is conducted to measure the influence of this ROI selection; this performance test immediately showed a perfect positive linear correlation between the total computational time and the image size. In Table 6.1, four test results are presented: one in which nothing is changed, one in which only the bottom part is removed, one in which the bottom and the sides are removed, and one in which all four sides are removed. At this point, only 50% of the picture is left. These four points are plotted in Figure 6.10 with the area reduction numbers used as the x-axis and the time reduction as the y-axis. Both reductions are expressed in percentages. In this picture, the calculated formula of the line is also stated. Reducing the ROI picture to 100% is theoretically impossible since most of the other algorithms will not work anymore.

6.5. Stereo Matching

The next block in line is stereo matching, this process was already discussed in section 2.3.5. In this section, a choice between the dense or sparse method and the working principles of the chosen algorithm are explained and implemented.

The choice for dense stereo matching was relatively straightforward. The most obvious reason to choose a sparse method is less computational power and less time. The cost is the loss of information, since it only uses features of edges and uses logic to calculate the depth in between. Combining that with the lack of features in the monotone landscape of the lunar surface will make this less than ideal. It is logical to choose a dense method since time is not constrained as the obstacle detection is far from being near to real-time.

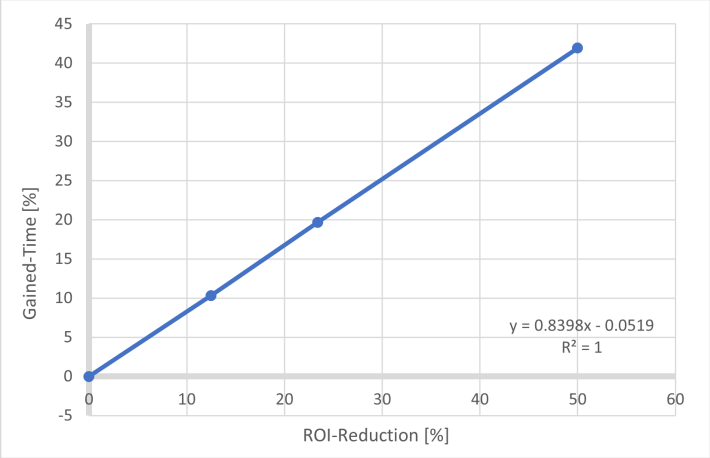


Figure 6.10: ROI Time Reduction

Description	Percentage removed	Time one revolution	Percentage Gained-Time
Full Picture	0	31	0
Removing bottom	12.5	27.8	10.32258
Removing sides	23.4	24.9	19.67742
Removing top using Canny	50	18	41.93548

Table 6.1: Small ROI Selection Influence Test

The dense stereo matching method scans both images line by line. Each pixel is matched using a similarity measurement method, there are multiple different methods developed such as Normalised Cross-Correlation (NCC), Sum of Squared Differences (SSD), Sum of Absolute Differences (SAD) and Semi-Global Matching (SGM) [17]. All these similarity measurement methods can be classed as local or global. Local methods focus purely on cost computation. The output is simple: the pixels with the lowest cost will be matched, resulting in a winner-take-all principle. However, in some cases, repetitive patterns or low-structured images make multiple match-winners [93]. For this reason, global methods are used to minimise the overall cost, also called energy.

$$E(d) = E_{data}(d) + \lambda E_{smooth}(d) \tag{6.1}$$

Where E_{data} is the energy function of all the stored disparity cost arrays, d. E_{smooth} is a smoothness term added, with all the stored smoothness assumptions of the algorithm.

The Semi-Global Block Matching (SGBM) method differentiates itself by not optimising the global cost of the whole image but by using blocks. The cost is minimised and smoothed in multiple directions (Top-Bottom, Top_Right-Bottom_Left, Left-Right etc.) while using these blocks. SGBM is a fast and accurate method proven during multiple studies [85] [107] [2]. One benefit is that it is implemented in OpenCV and proven to be applicable on the Moon's surface during the Chang'e 3 mission [56]. Hence, SGBM is considered to be a suitable choice.

There are several parameters to select and optimise, but figuring out the parameters, their function, and their used value is another. In Appendix E, a Graphical User Interface (GUI) is presented. With this GUI, parameters are iteratively changed to get a close-to-ground-truth result. The parameters are listed in Table 6.2 together with its determined preferred value.

Table 6.2: SGBM Used parameter list [78].

Name Parameter	**Used Value**	**Description**
minDisparity	0	Minimum disparity value, for if a picture after rectification is shifted a little.
numDisparities	96	Maximum disparity value (minus the minDisparity), maximum shift in pixels. Must be a modulus of 16.
blockSize	5	Size of the block used for energy minimisation. Normal Values are 3,5,7,9 or 11.
disp12MaxDiff	0	Maximum allowed amount of difference when comparing the left-right disparity. When zero it is disabled.
P1	$8 \cdot \text{blockSize}^2$	First parameter of the smoothing penalty function.
P2	$64 \cdot \text{blockSize}^2$	Second parameter of the smoothing penalty function.
uniquenessRatio	5	Percentage of whether second best or minimum option should be considered. Normally, this value is between 5-15.
speckleWindowSize	0	Speckle filtering window size value, disabled by setting to zero.
speckleRange	0	Speckle filtering range value, disabled by setting to zero.
preFilterCap	0	Truncation value of a pre-filtering function.
mode	mode_HH	Default mode is a mode where only five directions are considered. Mode_HH is a two-pass algorithm, considering eight directions.

As seen in the table, the pre-filter is disabled since, as discussed, the bilateral filter and the CLAHE filter are used for pre-filtering. Furthermore, the numDisparity together with the blockSize are found by using the GUI. This blockSize parameter affects the result of the stereo matching process significantly. In Figures 6.11 to 6.15, a couple of results of different block sizes are visible. The brighter the pixels the lager the shift or so called disparity value. The large block size provides reliable matches, where small window sizes provide accurate matches; but these are a little noisier. Because of time constraints, only one set of parameters was chosen, and no trade-off has been made between 'accurate' and 'reliable'. In the future, with proper precision and recall monitoring, a more optimal set of parameters can be found.

Most other values are set to their default values determined by the OpenCV community—for expamle, the two parameters of the smoothing penalty function. In a later stage, a changing penalty function based on imaging conditions would be more optimal, such as noise and illumination conditions [4]. Speckle filtering and left-right disparity checks are disabled as with the pre-filtering steps. Instead of this, a weighted least square filter is applied for post-processing the disparity output. This filter uses the original photo to align the photo edges with the disparity map contours, and therefore, it will develop the disparity values in low-confidence regions [30] [66].

Figure 6.11: SGBM result using a block size of 5 pixels

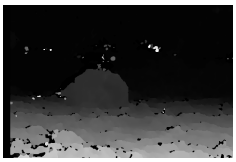


Figure 6.12: SGBM result using a block size of 7 pixels

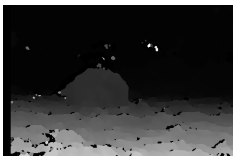


Figure 6.13: SGBM result using a block size of 9 pixels

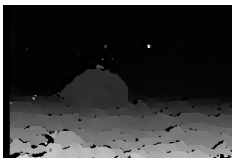


Figure 6.14: SGBM result using a block size of 11 pixels

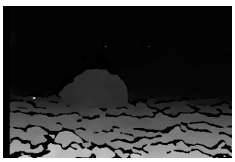


Figure 6.15: SGBM result using a block size of 13 pixels

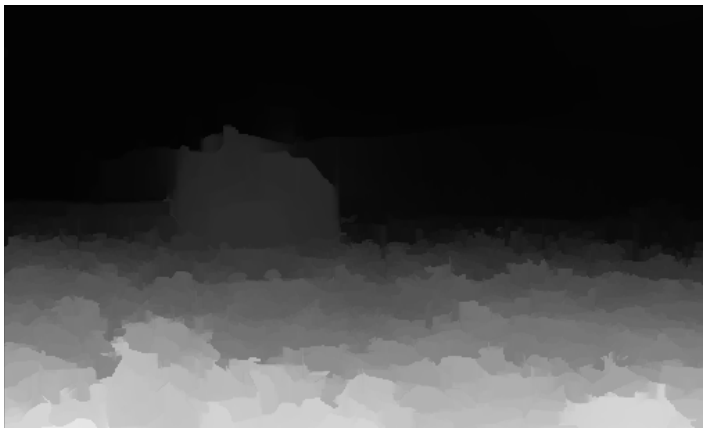


Figure 6.16: End result after stereo matching

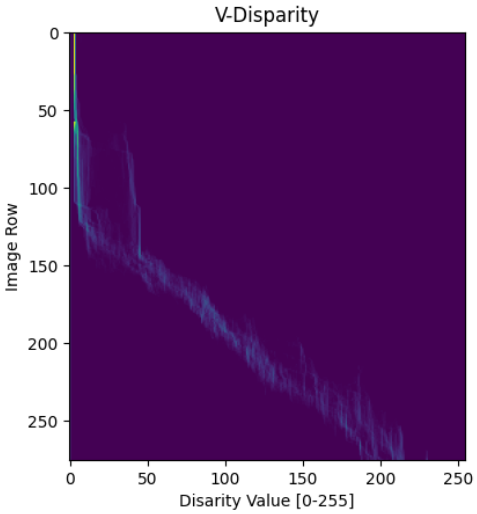


Figure 6.17: V-disparity result of figure 6.16

6.6. Ground-Plane Fitting and Deduction

The next block in line is ground-plane fitting and deduction. Most of the stereo vision research originates from the automotive industry. Urban scenes are very structured and the majority of the roads are flat. A flat-world assumption means that everything that sticks out is a possible obstacle, as mentioned by Manduchi [61]. With a more chaotic scene, obstacle detection using this assumption seems not the ideal solution.

Ground-plane detection has, however, also been used on 'off road' cases, and it could help an object detection algorithm to filter data points. Yiruo et al.[110] demonstrated, for example, that it is possible to use ground plane detection in such situations. An advanced v-disparity algorithm is applied to estimate the ground plane.

V-disparity is an algorithm proposed by Labayrade[52]; it uses the disparity output of the stereo matching algorithm and translates it to a v-disparity map, a histogram of the disparity in the horizontal rows. A histogram is a representation of the distribution of a set of values. In this case, the brighter the pixel the more those disparity values are presented in the row.

In Figure 6.17, the result is depicted and placed next to the previous result of the stereo matching algorithm. Clearly, in the rows where the rock is present, there are low ('dark') disparity values representing the sky and disparity values around the 50 indicating the rock. The only sloping line left is the ground, and to detect this, a line/curve fitting method is used. The two most popular methods in this research field are the Hough Transform and RANSAC.

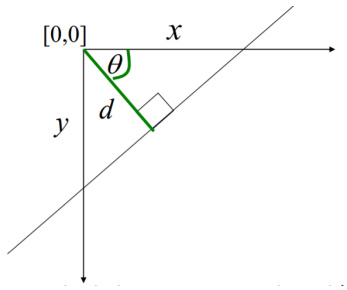


Figure 6.18: Polar Coordinates Representation [55]

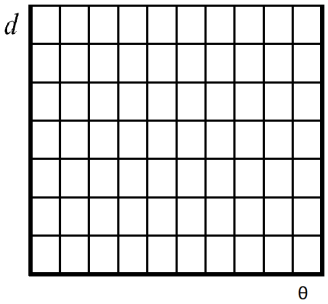


Figure 6.19: Hough Space [55]

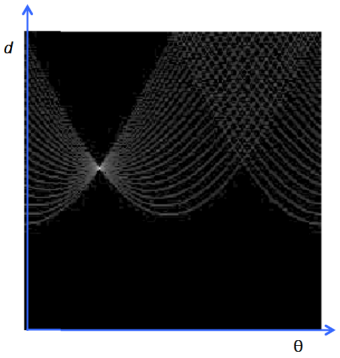


Figure 6.20: Hough Space filled with sinusoidal profile of data points on a line [91]

6.6.1. Hough

The Hough Transform [40] was invented to solve a repeatedly occurring computer vision problem; detecting simple shapes in a stack of datapoints. This stack contains a significant number of outliers, primarily because of noise or obscure parts. Hough introduced a voting structure, where each data point has a vote on the fit of a model. The model uses no more than two parameters. Therefore, polar coordinates are used to express a line with d—the shortest distance from a predetermined origin to the line—and θ—the angle between an axis and the line. In Figure 6.18, the polar coordinate system is visualised, representing the input for the corresponding equation for a line, stated in Equation 6.2. Finally, the Hough Transform algorithm is filling a so-called Hough space, visualised in Figure 6.19.

$$xcos(\theta) + ysin(\theta) = r \tag{6.2}$$

Every point has a corresponding sinusoidal profile in the Hough space representing all the possible lines through that point, also seen in Figure 6.20. The point in the Hough space where most lines intersect represents the best line for all the points. Adding some noise will cause the lines to not intersect at one point. Therefore, a grid is used, so the most dominant grid cell is chosen as the best line fit. This 'voting' system makes the Hough Transform robust to noise. However, the Hough lines are not entirely excluding outliers, which means the fit is not ideal.

6.6.2. Ransac

Very often, the RANdom SAmple Consensus (RANSAC) method is used to remove outliers to get a better position estimation. RANSAC is also used in various other applications. This iterative method seeks to fit a model through a set of data points. After its first fit, the algorithm labels some data points as inliers, and with these inliers, it tries again to fit a model to find new inliers. The algorithm stops after k iterations. According to Scaramuzza [92], k is calculated with Formula 6.3, where p is the chance of finding the suitable model. W is

the probability of choosing an inlier, and n is the number of all the possible points.

$$k = \frac{\log 1 - p}{\log 1 - w^n} \tag{6.3}$$

The algorithm is very flexible and can be used in any case with any dataset while not being very computationally heavy. The downside is that RANSAC requires knowledge about the data; and if there are too many outliers, the number of iterations will increase with a logarithmic scale.

Both the RANSAC and the Hough Transform methods were tested but were experiencing some issues. The Hough Transform provided multiple lines that could fit the ground. It was hard to come up with logic to decide which line to select. The line with the most steady result showed to be the line with the lowest positive derivative. The results are seen in Figures 6.21 and 6.22. Whereas in both figures, the Hough Transform output is always slightly detecting the wrong slope, the RANSAC method had problems with the high number of outliers in the dataset. Sometimes it performs well and sometimes poorly, as seen in both figures. This particular phenomenon was also mentioned by Li [55] and is entirely related to this high rate of outliers. A solution was to combine the two methods. First, with the Hough line detection method, an estimate of the ground is made. Then, the data is filtered with this estimate, reducing the rate of outliers. At this point, the RANSAC method is used to come up with a perfect fit for the ground.

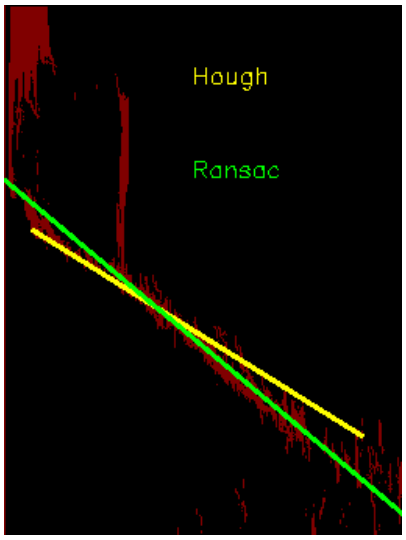


Figure 6.21: Ransac outperforming Hough

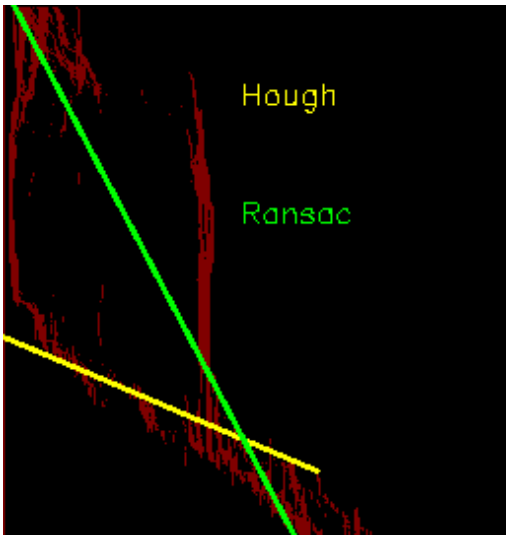


Figure 6.22: Hough outperforming Ransac

Now points of the ground can be selected and deducted. As seen in 6.21, the result of the ground line is a good fit for the bumpy and noisy ground. For selecting all the ground points, the area close to the line is used. The downside of this technique is that some of the information about the rock is also removed. In Figure 6.23 the selected ground points are visible. Here, the rock is partly selected as ground. The added logic to help detect more of the rock is seen in Figure 6.24. The v-disparity result of deducting the ground in a normal fashion is seen, after which the Hough Transform is used to detect vertical lines. These lines represent the obstacles in the scene. At this point, in an extension of this obstacle line, all the v-disparity points are added again, resulting in the fact that almost the whole obstacle could now be detected, as seen in Figure 6.25.

6.7. Clustering

Clusters are condensed areas of points in a graph, and they could be formed by two types of points—border points defining the border of the cluster and core points enclosed by the border points. Points that do not belong to any clusters are outliers. Clusters do not have fixed sizes or shapes and when the ground plane is deducted from the scene, multiple little clusters remain. For example, in Figure 6.26, where a rock cluster is depicted after the disparity map was converted to a 3D point cloud. A couple of things need to happen now

Figure 6.23: Subtracted Ground Result

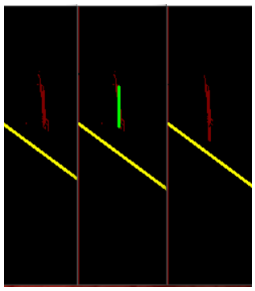


Figure 6.24: Extra Hough Line Detection Step

Figure 6.25: Subtracted Ground Result with Extra Hough Line Detection Step

before the closest rock can be sent to TRON. First, the different clusters need to be differentiated from each other and labelled, and other little clusters or outliers need to be removed.

Figure 6.26: 3D Obstacle Point Cloud

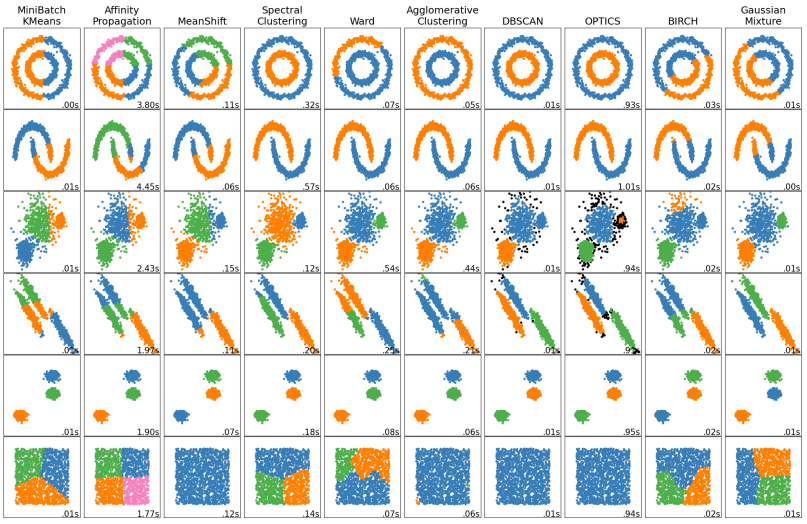


Figure 6.27: Comparison of Scikit-learn implementations of cluster algorithms [83]

Many algorithms are designed to perform detecting and labelling of clusters. Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is such an algorithm and is proposed by Ester et al. [27]. The DBSCAN algorithm uses a density-based approach: it looks for a group of points with a minimum density and a minimum size. However, when two clusters overlap, the DBSCAN starts struggling. Another solution is a centroid-based clustering algorithm like K-Mean, which was first proposed by Macqueen and further developed by others[31]. The algorithm looks for a pre-defined number (k) of clusters. Then it starts to attach points to the nearest cluster centroid and minimises the distance between each point and its centroid. These two clustering algorithms are the most widely used clustering methods, but there are more known clustering algorithms.

Scikit-learn is a library for Python which integrates many known machine learning algorithms while focusing on ease of use, performance, state of documentation, and consistency in the API [83]. The developers of Scikit-learn also compared all the clustering algorithms they implemented in their library, as can be seen in Figure 6.27. The cluster algorithm DBSCAN is also implemented in this Scikit-learn library. DBSCAN's performance is impressive as comparatively seen in the figure. Only once in the third example did DBSCAN have trouble differentiating a cluster. The time used for clustering is relatively low compared to the others. At the same time, DBSCAN is one of the few algorithms where no pre-definition of the number of clusters is needed.

Clustering in the OPAL algorithm, as mentioned, is needed for labelling and outlier filtering. When there is only one rock in the scene, this step seems unnecessary; but when multiple rocks exist, this step becomes useful. To demonstrate this a case with two rocks is processed in Figures 6.28 to 6.32. In Figure 6.28, we start with the ground being detected. In Figure 6.29 the ground together with the sky are deducted from the disparity matrix.

Figure 6.28: Two Rock Case - Ground Reduction Result

Figure 6.29: Two Rock Case - Filtered Disparity

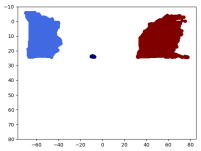


Figure 6.30: Two Rock Case - DBSCAN Result Front-View

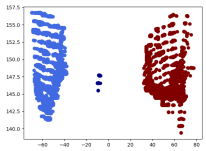


Figure 6.31: Two Rock Case - DBSCAN Result Top-View

Figure 6.32: Two Rock Case - Obstacle Detection Result

This disparity image is translated to a 3D point cloud. The top view of this point cloud, as seen in Figure 6.31, is then used as input for the DBSCAN algorithm. DBSCAN labels the two rocks and the outliers shown in the middle of the two rocks. How that is then translated to the result shown in Figure 6.32 is explained in the next two sections.

Figure 6.33: The result of OPAL with the Ground-Truth

6.8. Obstacle Classification

As defined in the requirements, an obstacle has a minimum height of 3 cm. Height definition sounds easy when the 3D points of the obstacle are known; however, it is not that simple. When a camera is in 3D space, as the ground has a slope, obstacles will include this slope. Rotating all the point around the x-axis with a rotation angle of the ground slope will make the height estimate more realistic. The peak of the cluster is now taken as the height of the obstacle.

This simple rule 'everything that sticks out is a possible obstacle', is the only rule applied for obstacle detection. In a later phase, more sophisticated methods for obstacle classification could be used. For example, the method described by Dunlop [24] used features to distinguish between rocks and the background. These features are, for example, shape, colour and texture.

A box representation is used to indicate the obstacle in the image and for algorithm performance measurement as discussed earlier in Section 3.5. However, sometimes, the obstacle is split into multiple parts by the DBSCAN. This splitting happens because the boundaries of the rock are sometimes tagged as belonging to a different cluster. This is because the disparity value is an integer value, and gaps could occur when translating

it back into 3D values. Overlapping bounding boxes are filtered out using a non-maximum suppression algorithm [7]. The final bounding box is seen in Figures 6.32 and 6.33.

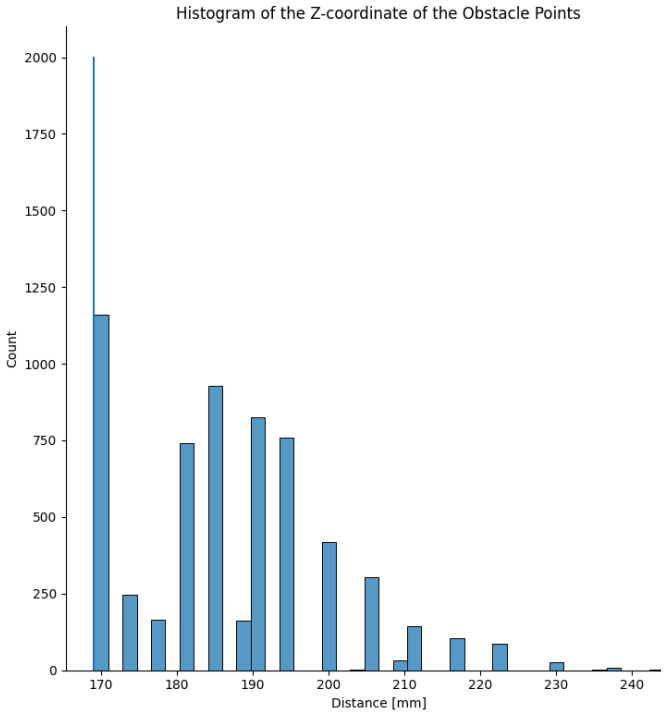


Figure 6.34: The histogram of the Z-Coordinate of the Obstacle Points

6.9. Distance Determination

Now that obstacles are classified as obstacles, the distances need to be determined. The distances were already determined when the disparity image was translated to a 3D point cloud using the triangulation method in Section 6.7. In Figure 6.34, the histogram of all the z coordinates of the obstacle points could be seen. This histogram is now used to detect where the obstacle starts from the camera's point of view.

If the number of obstacle points of a certain z-range is large enough, this point is taken as the distance from the obstacle to the camera. For example, in 6.34, there are around 1200 pixels that are 170 mm away. The threshold in the algorithm is set to 100 points. So, in this case the first distance value is taken as the final distance. At this point, all the distances are determined for each obstacle, and the obstacle that is closest could be communicated to TRON. This distance is added to the resulting figure seen in Figure 6.33.

7

Results

This chapter presents the results of the stereo vision obstacle detection system. At first, the measurement error is discussed. Furthermore, OPAL's performance is described over a case validation. At last, some general remarks were elaborated on, and the compliance of the requirements set in Chapter 4 is reviewed.

7.1. Measurement Accuracy

During the stereo vision obstacle-detection system development, a slight error was detected while the cameras were moving significantly. The hypothesis about this particular problem was based on the idea that the cameras were moving. There are two main reasons why this hypothesis could be correct. The first reason is an error in the design. During the design of the cam holder, it was decided that the camera module would be clamped. The camera module of the Raspberry Pi camera is tiny and is stuck with some tape to the PCB behind it. The camera mount of the test model clamps this tiny black box inside a frame which is more or less the same size. However, due to vibrations, the camera module shifts up and down in a direction not blocked by the casing. The other reason is the shear stress in the body because of the moving tripod principle or other movements of the body. This shear stress influences the stereo base, which is likely because the test model is just a piece of plastic and therefore not rigid.

Throughout testing, the system was calibrated multiple times with a chequerboard. The results of these calibrations are shown in Appendix C. Usually, these chequerboard calibrations also contain errors. After triangulation, using the calculated camera parameters, the corners detected by a corner detection algorithm do not perfectly fit each other. In other words, the projection of all the calibration patterns does not match. The algorithm is specifically trying to minimise this, which is also called a re-projection error.

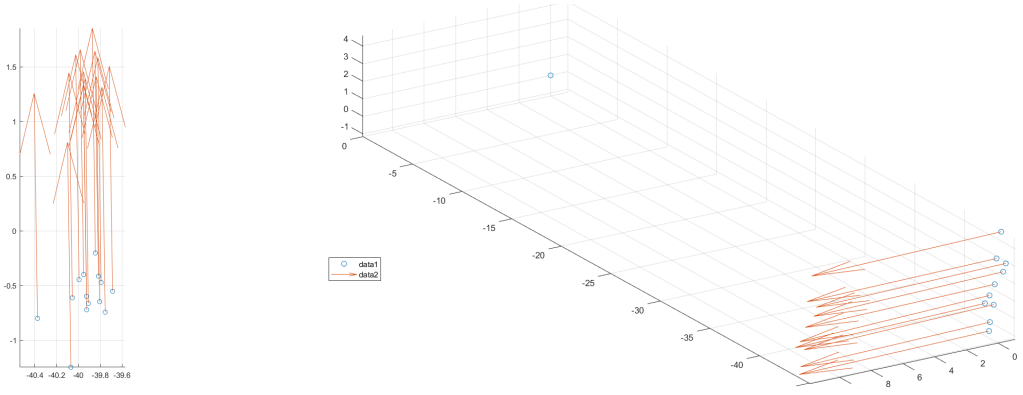


Figure 7.1: Visualisation of the Calibration Error - Detailed top view and isometric Overview

The results of the calibrations in Appendix C are put together in a graph in Figure 7.1 to visualise the influence

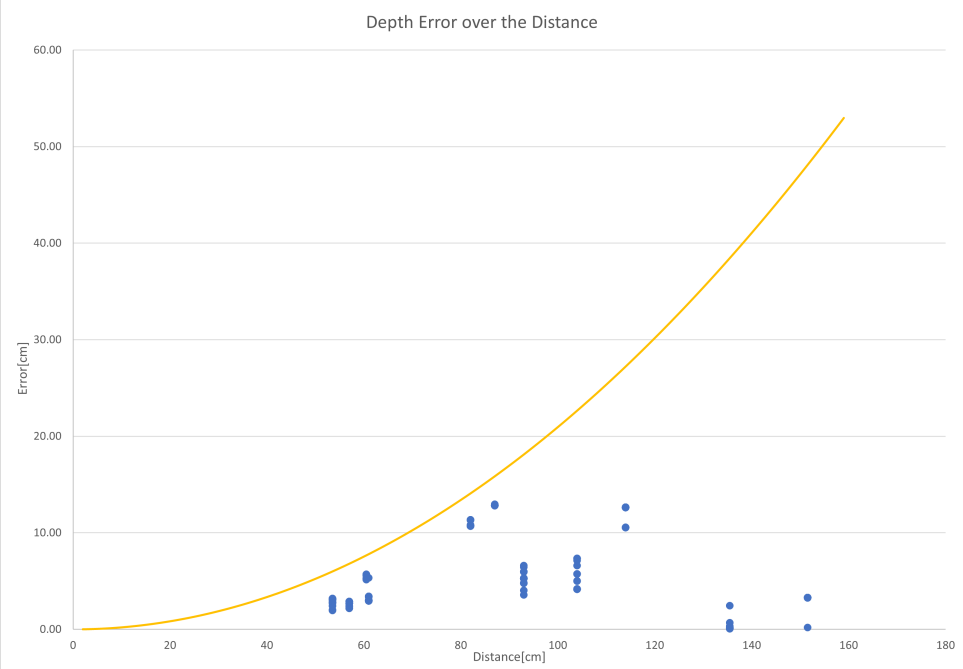


Figure 7.2: Theoretical depth error over the distance, together with the static output of OPAL

of this re-projection error on the final camera model. As seen, the camera models vary in translation and rotation. The standard deviation in the x-, y-, and z-axis direction is 0.17 mm, 0.24 mm, and 1.8423 mm. The x- and y-axis standard deviations are negligible, while the standard deviation for the rotation around the z-axis is 0.26 degree. These fluctuations are also seen in the top-view plot. The rotation variation around the z-axis will probably correlate to the variation in translation in x and y.

Besides this re-projection error and moving stereo-base, the discretisation, as explained in Section 2.3.7 and the section above, will influence the depth estimation. The triangulation formula 7.1 mentioned (Section 2.3.7) describes the relationship between the error in the depth ΔP_z and the distance P_z. Furthermore, the focal length of the cameras f, the distance between the cameras b, and the pixel width Δx are used.

$$\Delta P_{z,triangulation} = \frac{\sqrt{2}\Delta x p_z^2}{fb} \tag{7.1}$$

Furthermore, the error created by the stereo matching process also affects the resulting accuracy of the depth estimation. Every little error in ΔP_d, which represents the disparity error, will accumulate due to the stereo geometry. For the disparity error, the resulting depth error is caught in Equation 7.2 [50]. According to the Middlebury stereo evaluation database, the average Root Mean Squared Error (RMSE) of the disparity, which is expressed in the number of pixels, is 41.7 pixels [18]. Therfore, the disparity error, ΔP_d, could also be expressed as $41.7\Delta x$. The errors in camera calibration, including the lens distortions and camera alignment errors discussed at the beginning of this section, affect the disparity error. An additional study will need to be carried out in order to resolve this process. For now, it is assumed that this relationship is included in the number extracted from the Middlebury dataset.

$$\Delta P_{z,disparity} = \frac{p_z^2}{fb} \cdot \Delta P_d \tag{7.2}$$

Adding these two errors together will result in Equation 7.3. This formula is plotted in Figure 7.2 using for Δx, f, and b the values 1.4 μm, 3.6 mm and, 40 mm respectively.

$$\Delta P_z = \frac{\left(41.7 + \sqrt{2}\right)\Delta x p_z^2}{fb} \tag{7.3}$$

Moreover, in this graph in Figure 7.2, data points are visualised. The data points visualise the output of OPAL. During the test cases discussed in the following sections, the rover is stopped multiple times. Within these stops, the distance to the rock is measured with a tape measure. Clips of the footage belonging to the measurement are then processed using OPAL. The error of these data points are plotted in the graph in Figure 7.2. As seen, all the data points measured are within the maximum expected error.

7.2. Case Validation

In this section three cases were used to validate OPAL. To reduce the amount of time needed to process all three video's, only every second frame of the footage of each case is used. During each case an overview is provided first, afterwards the performances are elaborated on.

Figure 7.3: Case 1: Scene Capture

7.2.1. Case 1

In this first case, Bars is walking towards one rock, obstacle two. This rock is specified as a flat and wide rock. Behind the rock, there is a crater ridge. The walking path does not deviate much in this case. The rock is, therefore, observable in the middle of the camera for almost the entire video. Sometimes during walking, the rock sides disappear from the scene, but most of the rock stays visible. During the walking phase, the rover is stopped four times to measure the distance to the rock. Together with the beginning, where the author is also performing calibration and an initial measurement, these parts are clipped from the video and processed by the algorithm. In Figure 7.3, a snapshot of the video feed to the algorithm is shown. Here the processed rock is visible.

Algorithm Performance

Processing the cut video feed took on the Intel NUC server around 120 minutes to process 983 frames; on average this is around 7.3 seconds per frame, including overhead such as creating an output video, creating an output log, and other operations. This output log contains, for example, the frame number, the distance to the closest obstacle, and determined IOU.

In figure 7.4, these three values of every processed frame are visualised in a graph by dots. Each dot could be green or red, representing if detection is a true or a false positive as determined by this IOU. In the graph, a number of other items are presented. There is the expected path represented by the blue line. The striped lines above and below the blue line are the minimum and maximum theoretical error, as explained in Section 2.3.7. Furthermore, the three red lines represent the determined requirements as explained in Section 4.2.1. The first striped line where is the line the rover should stop for the rock and start acting. Between the last two red lines, the danger zone starts. If an obstacle passes one of these lines, there is a possibility the obstacle becomes an untraceable hazard as explained in Section 4.2.1. Finally, the values of precision and recall are stated in the red text box. These indicate the performance of the algorithm as explained in Section 3.5.

Figure 7.4: Case 1: Result Graph

A couple of remarks can be made about the algorithm. Clearly, the obstacle in the scene is detected. However, the algorithm is not detecting the whole rock from the beginning. The distance detection, on the other hand, is performing within the error margin and doing so fairly constantly. It is important to note is that the estimated walking path is just an estimate and the horizontal lines are the actual measured distances. The rock was correctly detected before the rocks hit the danger zone, which is the most crucial criterion.

Figure 7.5: Case 1: Problem Rock seen as Ground

The rock was as discussed false detected in the beginning. At this point, This is not really a problem since the rock is still far from the Lunar Zebro. In Figure 7.5, the ground-plane deduction is shown in the v-disparity graphs. The v-disparity was the histogram of the disparity per row, and explained in Section 6.6. First, the full v-disparity graph together with the detected ground-plane line are shown. Afterwards, the ground-plane disparity values are removed. What is left is a tiny line. This line was shown to be too small to be detected by the Hough Transform algorithm. Therefore, the rest of the obstacle points are not recovered, and the obstacle stays halved.

At the end of this case, once again some red dots appear in the last part of the graph in Figure 7.4. The reason

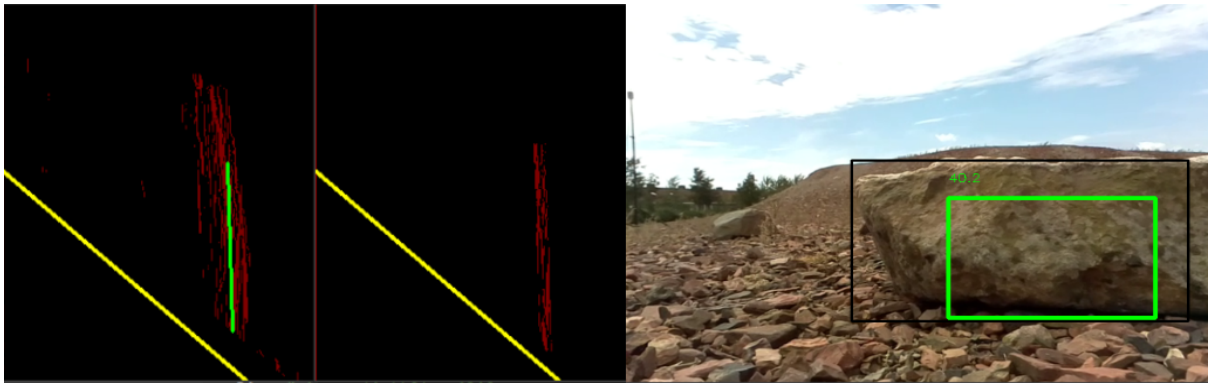


Figure 7.6: Case 2: Problem Partial Detected Rock

for this is seen in Figure 7.6 where the output v-disparity is seen when detecting an obstacle in the v-disparity domain. In this domain, the spread of the disparity increases when an obstacle gets closer to the camera. A small area around the detected line is seen as part of the obstacle in the algorithm. The rest of the disparity values are filtered. Therefore clearly, only a part of the obstacle is detected in the resulting figure located on the right-side of Figure 7.6. This is not effecting the obstacle detection in the critical zone, since still a representative distance is measured.

Figure 7.7: Case 2: Scene Capture

7.2.2. Case 2

In this second case, Bars is again walking towards one rock, obstacle one. The walking path deviates a little in this case, which results in the rock almost leaving the field of view. Like the other case, the rover stops four times during the walking phase for measuring the distance to the rock. These parts are clipped from the video, together with the beginning, and the video is processed by the algorithm. In Figure 7.7, a snapshot of the video feed to the algorithm is shown.

Algorithm Performance

Processing this case took 148.5 minutes, with 1015 frames processed for an average of 8.8 seconds per frame. The precision here is higher than in the first case, mainly because the ground was not correctly deducted initially in the first case. In this case, the false positives are more spread out over the whole length of the test case. This false detection has several reasons; instead of too much ground-plane deduction, there was rather too little ground plane deduction. The other problem, in this case, was that the algorithm suffered when the rock disappears from the field of view. Matching and detection were therefore difficult to be performed.

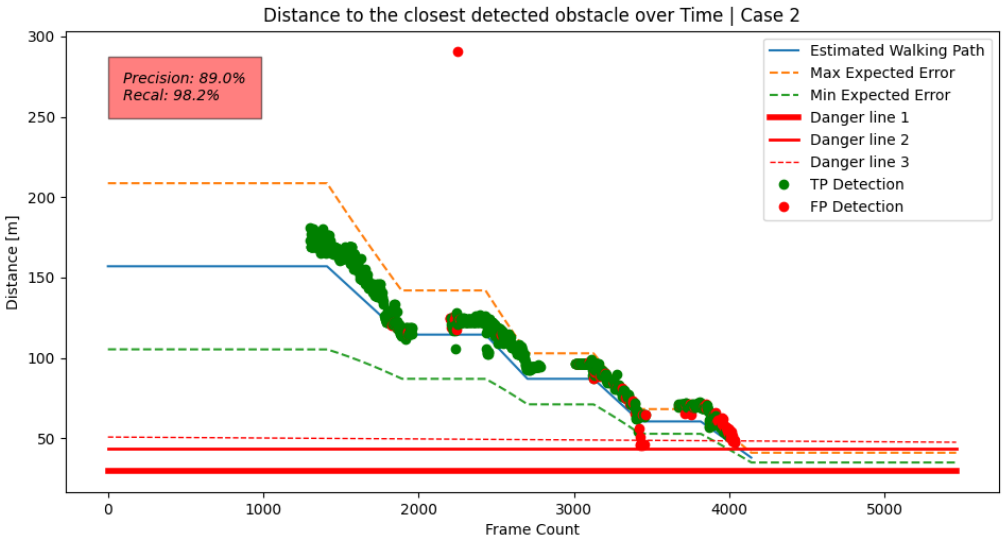


Figure 7.8: Case 2: Result Graph

The first problem—the red dots around the 2000 frame count in the graph in Figure 7.8—lies in the ground-truth box definite by the CSRT algorithm, which tracks the obstacle. The detail of this algorithm are discussed in Section 3.5. Parts of the rock are in the field of view of the left camera but not in the field of view of the right camera. Causing that part to be subtracted from the box. OPAL is only detecting a small part of the rock where the tracked box is still using the upper and the lower limit of the old box, which results in a false positive since the IOU is now no longer matching. The result is depicted in Figure 7.9.

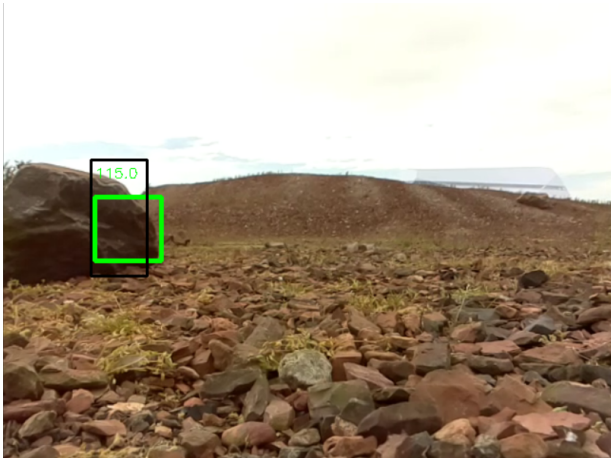


Figure 7.9: Case 2: Problem Bounding Box at the Side

The second significant thing to note on the graph containing the result is the upper red dot. This red dot represents one of the few complete misses of the algorithm. There are two reasons for this. First, something is happening in the clouds, which makes the sky 'light up' in the disparity map. The scene and the resulting disparity map could be seen in Figure 7.10. Secondly, the wrong obstacle was detected in the v-disparity and wrongly filtered. This process is shown in Figure 7.11 on the left side. The left mask and the resulting bounding boxes are seen on the right side in this figure. The problem can be fixed by optimising the code but also by adding a filter which includes previous processed results. Since the surface of the Moon is static landscape, this is a validate option.

The next issue is the bulge at the end of the third walking phase in the result graph. Figures 7.12 and 7.13 helps

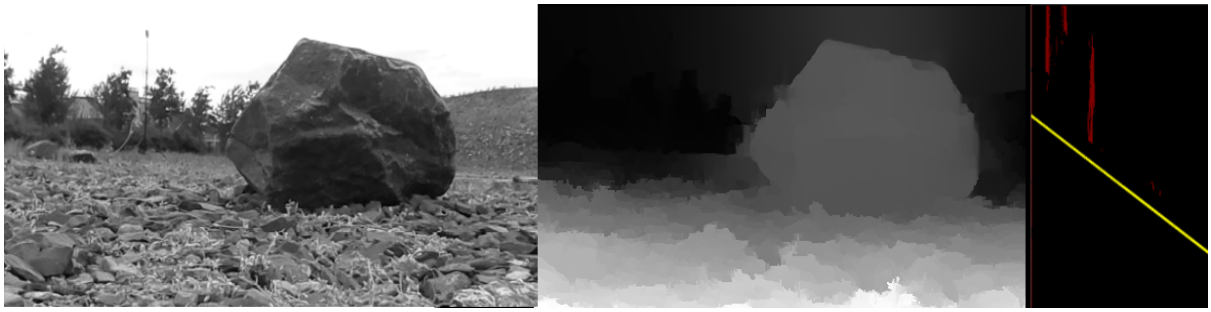


Figure 7.10: Case 2: Problem Detected Sky Part 1

Figure 7.11: Case 2: Problem Detected Sky Part 2

visualising this problem. There are three clusters detected, as seen on the left side of the first figure. This graph is the top view on the 3D point cloud after ground-plane deduction. The red and green ones are little bumps and not seen as obstacles. The light blue cluster is the represented rock in the scene. The rock contours can be seen, but there is a little bulge and some ground points on the bottom of the cluster. In this same figure, on the right side, it can be clearly seen that the ground deduction was not ideal, and less ground was deducted. This same ground was causing points closer to the rover to be included in the cluster. These points were then taken into account in the histogram and caused the algorithm to think the obstacle was closer than it was. This histogram of the distances of the obstacle cluster is seen in the second figure on the left side. On the right side, the result of this case is seen.

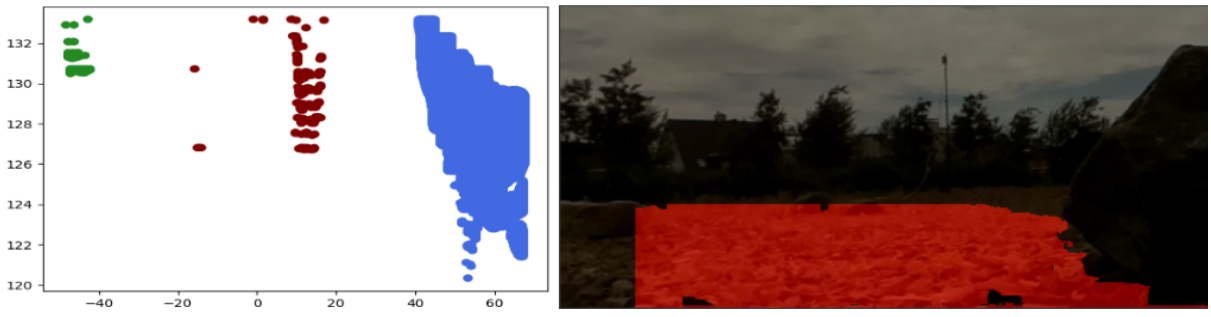


Figure 7.12: Case 2: Problem Ground in Obstacle Cluster Part 1

Figure 7.13: Case 2: Problem Ground in Obstacle Cluster Part 2

Figure 7.14: Case 2: Problem Partial Detected Obstacle

In the end, the same problem occurred as in the first case. There is too much rock filtered in the disparity domain because the disparity range increases when the obstacle gets closer to the camera. On the left side of Figure 7.14, the v-disparity of the scene is apparent. Only the area around the green line is left after filtering the obstacle from the disparity, which result in the green bounding box seen in the picture on the right side of the figure.

Figure 7.15: Case 3: Scene Capture

7.2.3. Case 3

In this last case, Bars is walking towards two rocks now, obstacle one and obstacle four. These rocks are placed behind each other with a distance of 64 cm between the two rocks. The walking path is relatively straight without much deviation. Now, however, there are only two measurements performed: one at the beginning and one at the end. So only the beginning and the end are clipped from the processed video. In Figure 7.15 a snapshot of the video feed to the algorithm is shown.

Algorithm Performance

The time spent by the algorithm in the third case is around 73.5 minutes for 601 frames, which is an average of 7.3 seconds per frame. Compared to the first two test cases, the precision and recall are far lower. The precision is lower because many false positives are detected. The recall becomes lower because there are fewer true positives than the other test cases. Therefore, the influence of the detected false negatives increases. In the graph in Figure 7.16, the reason for this amount of false positives can intuitively be concluded. As may be inferred from the graph, the algorithm is switching between the two rocks. While it sometimes succeeds in

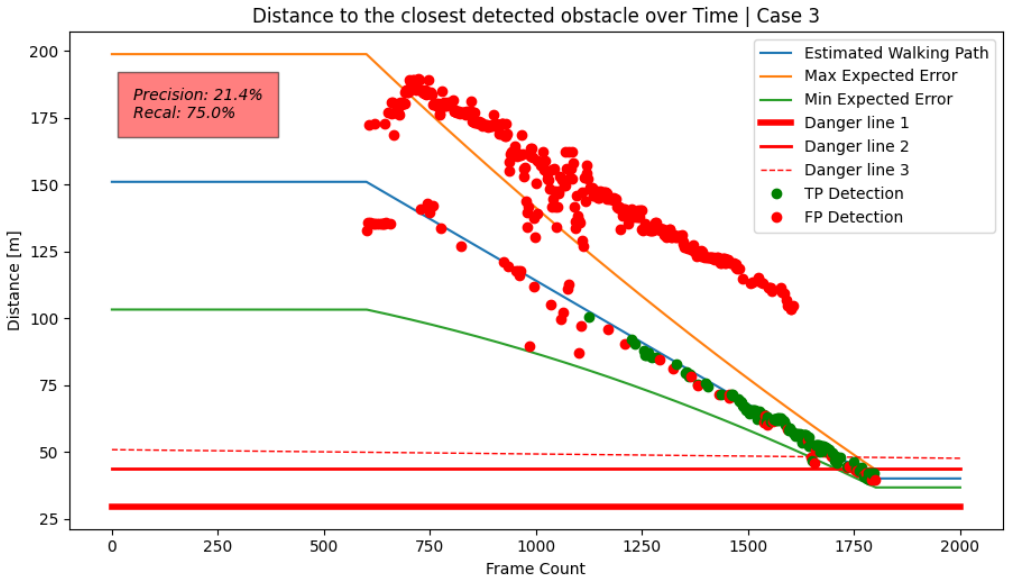


Figure 7.16: Case 3: Result Graph

detecting the rock, it is also struggling and detecting the wrong one. As noticed, the rock is properly detected when the rock is entering the crucial distance.

In the disparity domain, the spread disparity values make the Ransac line's slope fit poorly, resulting in a rock deducted so much that the only rock left is the second one. In Figure 7.17 on the left side the wide ground disparity range is visible. In the middle of the figure, too much of the rock in front is highlighted as ground. As a result, the resulting bounding box around the wrong rock is seen on the right side.

Figure 7.17: Case 3: Wrong Detected Obstacle

7.3. General Remarks

During all these test cases, there are moments where the ground is not correctly detected. When the Lunar Zebro is operating normally (i.e., walking in a usual manner over the surface), the angle between the camera's optical axis and the ground is expected to be within a particular range. The extraordinary, detected ground planes are marked as unprocessable for the remainder of the algorithm. The other false-negative possibility is that a wrong ground-plane fitting could lead to too much ground-plane deduction. This could happen when walking towards smaller rocks. The last case showed that this problem disappears when the obstacle becomes

closer, and the rock is detected again. The high percentages of the recall parameter suggest that the number of false negatives is acceptable.

The time spend by the algorithm was also considered acceptable. For example, case 2 was run using the GitLab runner on Intel NUC with a 2.90 GHz processor inside; this took around 8.8 seconds per frame. The actual processor is around 500 Mhz. The closest hardware available is a Raspberry Pi with a processor of 1500 Mhz. The time the algorithm took on this platform was around 23.5 seconds on a single core. When interpolating this to a 500 Mhz processor, the algorithm will take the estimated time of 34.2 seconds for processing pictures in case 2, which is within the time-limit.

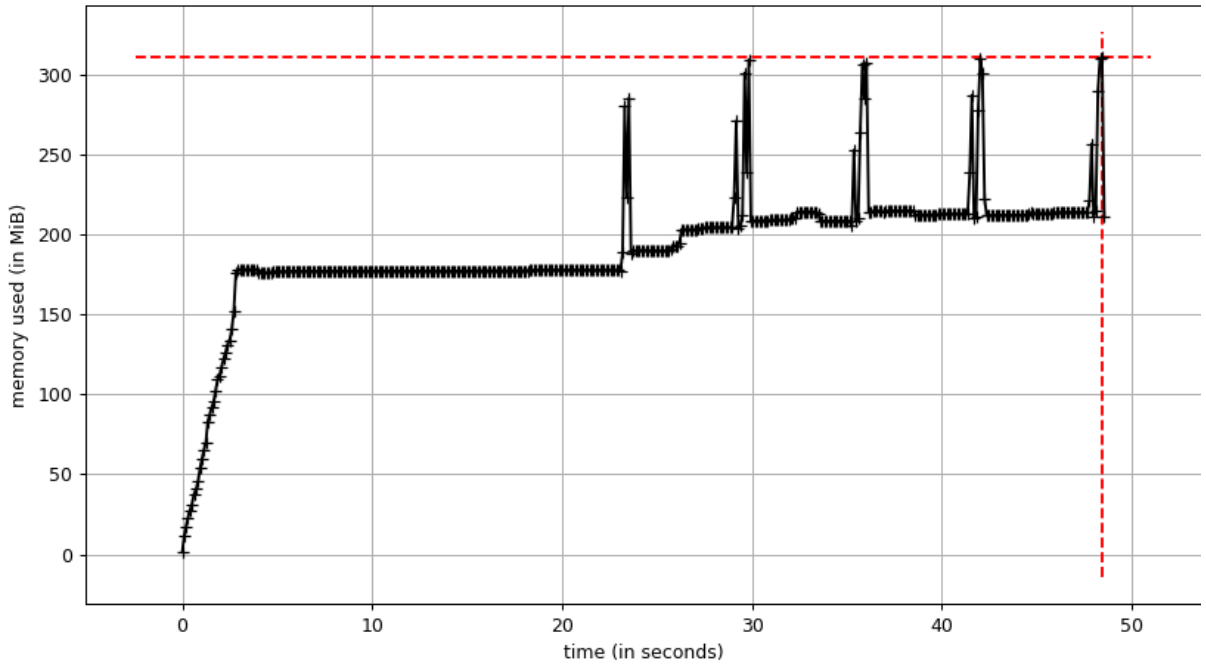


Figure 7.18: Memory Usage of OPAL

For checking the memory usage of the algorithm, a memory profiler is used [29]. This memory profiler can monitor the memory consumption of Python scripts. (See the graph in Figure 7.18) The memory usage is plotted over time. The algorithm runs five frames in case two. The initialisation phase can be identified. After this, the video fast-forwards to the walking phase, and then finally the frames are processed. A maximum of around the 310 MiB is reached, which is around 325 MB.

To conclude this section, the compliance of OPAL for each requirement is checked in Table 7.1. The colours indicate compliance, where green is full compliance and yellow is almost compliant. The requirements which were not compliant were already subtracted in Section 4.3. Furthermore, there is a rationale provided as to why these requirements are compliant or not.

Table 7.1: The compliance on the requirements of OPAL

ID	Requirement	Validation Method	Compliance Rationale
OPAL-INT-04	OPAL shall have as input the images produced by two SHRIMP cameras with a stereo base of 40 mm, a resolution of 640 x 480 pixels and a field of view of 65 degrees.	I	OPAL currently uses two Raspberry Pi cameras, but the used resolution of the camera is the same as the SHRIMP camera.
OPAL-INT-05	OPAL shall be able to be run on the ZPU, which has 500 MHz processor and is Linux based.	D	In theory, it should be able to run on the ZPU in the current state; however, this is not tested.
OPAL-INT-06	OPAL shall not use more than the 400MB of RAM.	D	This is demonstrated; OPAL is not using more than 300 MB. This could, however, vary from the final implementation on the ZPU.
OPAL-FUN-01	OPAL shall be calibrated with a re-projection error lower than 0.2 pixels.	D	The re-projection error of the used calibration is 0.15 pixels.
OPAL-FUN-02	OPAL shall limit the contrast with a maximum change in intensity to limit the influence of lighting on the algorithm.	D	This is implemented in the algorithm; it still needs to be tested or demonstrated if this also decreases the influence on the lighting.
OPAL-FUN-03	OPAL shall detect the presences of rocks larger than 30 mm in front of the Lunar Zebro.	T	The rocks presented in the testcases were all larger than 30 mm and detected by OPAL. However, more tests are required to gather data of more rocks with different sizes.
OPAL-FUN-04	OPAL shall detect hazards within the detection area pointed out by Figure 4.7.	T	In the three presented cases, the algorithm detected the rocks in the critical error range.
OPAL-FUN-05	OPAL shall determine the distance to every hazard with an error less than the theoretical error curvature.	T	Most of the times, the measure error falls within the theoretical error range, especially in the static test in the graph in Figure 7.2. However, the algorithm still measured outliers which were outside this error range.
OPAL-FUN-06	OPAL shall not exceed the time limit of 73 seconds to process a frame.	T	The actual time per frame used is not tested on the actual hardware; however, it seems most likely this requirement will pass when testing this (and definitely when OPAL is also translated to C++).

8

Conclusion, Discussion and Recommendations

In this chapter, a conclusion, a discussion and some further recommendation are disclosed. Together, these sections concentrate on the academic value of this thesis.

8.1. Conclusion

The Lunar Zebro is a small six-legged robot. It has the potential of becoming a swarming robot carrying out objectives like exploring planetary surfaces or building space antennas. With OPAL, a new step towards an autonomous navigation system, which is required when having a swarm of multiple entities, is made. This study showed that this initial iteration of OPAL could detect rocks and determine their absolute distance to the rover's low perspective cameras. Furthermore, this study formed the basis for more research into OPAL and its testing platform.

It took six steps to achieve this goal. At first, requirements were defined for the final OPAL algorithm, which is the input for the verification process of OPAL. Hereafter, a test model called Bars was successfully design and build. This test model is using, besides Lunar Zebro hardware, mostly Commercial Off-The-Shelf hardware, which is related to the Lunar Zebro hardware. Furthermore, Bars is optimised for easy operation on testing days. The tests of Bars were conducted at a location containing a Mars-like surface. The test facility had many similarities with the lunar landscape, like the grainy surface and rocks. There is a lack of monotone rough terrain test locations in the Netherlands and due to the Covid-19 period it was not possible to go to another (bigger) test facility as, for example, at the European Space Agency (ESA) in Leiden. Therefore this test space was considered sufficient for this pilot study. At this test facility, camera footage from Bars was captured.

With this footage, an algorithm was developed using primarily open-source libraries. A specified architecture set the order of the used algorithms of these libraries. Some added logic was required to connect and fulfil the implementation of these algorithms into OPAL. Hereafter, a pipeline on a server was created to run multiple test cases on the created test data and this established the results. One early conclusion of the results is that rocks are identified, and absolute distances were determined. In the cases where only one rock was present in the scene, OPAL showed the capability to have high precision and high recall detection. Where the Precision was the probability, a detected box is a real obstacle and the recall the probability that all obstacles are detected. The performance dropped when an extra rock was present, but OPAL still could detect the closest rock at the critical distance. These results were further validated using the set of requirements defined at the start of this project. OPAL is compliant or partially compliant to most of the requirements. An overview is provided in table 7.1. OPAL was not fully compliant with all of the requirements, which is further elaborated on in the discussion and recommendation sections.

This research is a success if it could answer the research questions set at the start of this thesis. Each research question is evaluated separately:

R1 - How could obstacles and their absolute distance to the rover be detected and how accurate does this distance need to be?
Obstacles and their relative distances could be detected using the so called disparity map. The disparity map is obtained from performing stereo matching on footage of a stereo camera. When translating the disparity to V-disparity which is a histogram of the disparity per row, a slanted line could be seen. This slanted line represents the ground, when assuming the ground is nearly flat. In this same V-disparity, obstacles are represented by vertical lines. In this thesis, obstacles are mostly detected in the V-disparity domain. Furthermore, after translating the remaining disparity map to a 3D map, a clustering algorithm is used for clustering and labeling obstacles in the scene. From this labeled obstacle clusters the distance is determined. In Chapter 4 all the requirements for the interfaces of OPAL and the performance requirements are stated. One of the performance parameters is the stopping distance before a rock becomes an untraceable hazard. Along with the stopping distance, the safety margin is discussed by using the worst case scenario. In this scenario the Zebro takes a 14cm leap. So actually this stopping distance is shifted 14cm backwards. At this distance the accuracy need to be at least 7 cm to even make it possible to detect that it is in this "Danger Zone". The final stopping distance is then evaluated as this shifted stopping distance plus the final determined theoretical error, given in equation 7.3.

R2 - How could the obstacle detection system be implemented to run on the Lunar Zebro and detect obstacles and their absolute distance to the rover from a low perspective, and how accurate can it determine distance?
An architecture helped structuring all the software blocks needed to gain a result. These software blocks consist mainly of open-source libraries. After this structure was created, some extra logic was added to connect all the different blocks. All the steps taken are described in Chapter 6 where the design and the integration steps are elaborated on. For the integration step, it was very important to have the test data available that was created by Bars. There is still room for more improvement, maturation and optimisation but the algorithm showed, as described, some decent results. The low perspective remains a challenge, because when obstacles disappear from the field of view, they become hard to detect and track. Another challenge is posed by large rocks as they could block the whole field of view when nearby.

As explained, the accuracy on which the OPAL could operate is defined by adding the triangulation error to all the digital errors condensed into the disparity error created by the stereo matching algorithm. The error curve is shown in figure 7.2 and is near the "Danger Zone" around the 4.8 cm.

R3 - How to validate a low positioned stereo vision obstacle detection system for a planetary rover in the Netherlands?
By design and building a test model, tests could be conducted at suitable test locations. Here, some actual operations on the surface of the Moon could be simulated. The recording of this on-board footage will create a helpful data set. If the rover's position is recorded as well, the data will become helpful for validation. During this thesis, a measuring tape is used to secure the location at some points. This method is not fully reliable since the rover's location is estimated during the walking phase. Another measurement method, where the real-time position is measured, could have generated a more reliable data set. Unfortunately, OPAL is never tested and run on the actual hardware of the Lunar Zebro because of resource restrictions. So for that part no validation is conducted.

All in all, this thesis is fulfilling its objective and answering its research questions. Together Bars and the initial iteration of OPAL are opening more research and development opportunities in the near future. With the development of Bars, more advanced testing can be conducted at Decos, or other exciting locations here on Earth. Furthermore, Bars could be equipped with SHRIMP so this footage can be added to the data set. Both can lead to better verification and validation of OPAL. On the other hand, the development of OPAL is, as mentioned, far from complete. This study exposes its opportunities and challenges, which could be a starting point for optimisations or other approaches.

8.2. Discussion

As mentioned in the conclusion there is room for more improvement, maturation and optimisation on several levels. In this chapter these levels are discussed.

The interface and platform adaptions, because of time and resource restrictions, should be emphasised. First, the final version of OPAL needs to be written in C++ so it can be integrated in the master controlling program TRON, which is, unfortunately, one of its primary interfaces. Furthermore, OPAL is not run on any of Lunar Zebro's ultimate hardware. For example, OPAL is not running on the Hyperion OBC and is not using a stereo vision system containing SHRIMP cameras.

Another possible improvement could be; the optimisation of OPAL components. One of those components is the detection of obstacles in the V-disparity domain. In for example test case two, an outlier occurred; a completely wrong obstacle was detected. This happened because that one obstacle was confused with the ground, and helpful information was therefore filtered. In the other case, the obstacle had a large spread in the V-disparity domain where the filtering area was too small. This resulted in the detection of only a part of the obstacle. In the first issue, obstacles became detectable again in the range where detection was becoming critical. The other issue did not, in practice, influence the distance-detection result, which is the most critical function of OPAL.

Both previous issues could be solved by changing the method and the timing of the ground-plane deduction. When fitting a line model through the ground points, the ground plane does not always have the right inclination angle. Therefore, sometimes too much rock is seen as ground or too little ground is deducted. Besides optimising by fitting a line through the ground points, a potential solution could be to detect all the vertical lines before deducting the ground. The only problem here could be that some ground patches are detected as obstacles. For example, in test case 3 (Figure 7.17), many vertical lines could be seen in the ground points. If they are too close to the obstacle, these points could end up in the obstacle cluster. The other issues (such as containing the large spread in the V-disparity domain) could be solved by a dynamic filtering range—the closer to the obstacles, the larger the disparity range.

A lot of OPAL parameters are now either adopted from the literature or set to a fixed value. These parameters could be reconsidered when developing an advanced version of OPAL. The calculated recall and precision could help to optimise these values.

In addition, at the level of validation, the CSRT algorithm used for tracking the ground-truth of the obstacle was not perfect. The obstacle is selected manually and is also manually corrected for the little drift. Hence, the obtained ground truth of the CSRT algorithm is not the real ground truth. As discussed in the result section, the bounding box is corrected, because sometimes the majority of rock is not visible by the stereo camera. However, the height of the bounding box should also be adapted because the rock is on the side not as high. This leads to erroneous false positives. Keeping the obstacle centred in the field of view could be a solution, which could minimise this side problem. Forms of odometry, either visual or with sensors like an IMU, could detect walking path fluctuations. A straight path towards obstacles could be obtained while having a locomotion system that could correct for these slight deviations. When the rock is always straight in the field of view, the risk of rocks becoming a hazard is minimised. On second thought, this is only necessary when the rock is close, since rocks far away will not become hazards yet. Furthermore, the next conducted test should include more test cases like craters or large inclined zones. OPAL should be redesigned to be able to detect these as well. Also important for the validation is better position data. For example, a good solution for this could be to use a Real-Time Kinematic positioning (RTK), which is a system designed to enhance GPS data.

The last level of improvement is the processing pipeline. Currently, the processing pipeline is memoryless which means that every image is handled as an individual entity. However, since the lunar surface is a very static environment, obstacles could easily be tracked over multiple frames. Outliers as shown in the results could then easily be filtered when looking at a sequence of images.

To conclude, all these levels of improvement seem solvable and have definitely been taken into account during some follow up study.

8.3. Recommendations

Some further recommendations are stated in this section. These recommendations origin from the development of OPAL and Bars and are including some remarks to pay attention to, or other techniques which could be included in, for example, the follow up study.

The rover and its navigation system must be aware of the extreme lunar conditions. The radiation in the lunar environment is harsh and dangerous. The OPAL software needs to be revised according to the standards that the TRON engineer set. The difference between day and night temperatures are extreme, but also the difference between sun and shadow. Within the team, it must be decided if shadows are hazards. The temperature drop in the shadow will be very high, which could harm the electronics. In addition to that, seeing in shadows is expected to be very difficult due to the harshness of the shadows. This could lead to false-positives or negatives in the obstacle-detection system. At last, the rover must be aware of the lunar dust. Due to the low conductivity, the dust is charged with an electrostatic charge and wants to stick to other surfaces. The dust could cover the lens of the camera. Again, the obstacle-detection system would not be able to observe possible danger.

Furthermore, the stereo base distance could be optimised. This thesis showed that the error becomes smaller if the stereo bases are made wider. The disadvantage is that it is harder to detect rocks closer to the rover, since the overlap of both cameras becomes smaller at closer points. This could be solved by making use of more cameras. For example, three cameras could improve the detection range and increase the camera system's redundancy in case of a single camera failure. With three cameras, comes three possible stereo vision couples could be used. With four cameras, this number increases to six. Now, the cameras are glued or clamped to their casing. The team should make sure that the different thermal expansion coefficients of the different mounting materials will not interfere with each other. Otherwise, a stereo setup that is calibrated on earth will not be calibrated for the lunar surface. Also, the stereo base rigidity should be increased with making the stereo base more rigid. Decoupling the camera system from the mechanical stress created by the body should reduce the disturbances and the calibration differences.

A couple of times, the rover flipped when trying to walk over an obstacle. The original design of the R-Hex robot of Boston Dynamics contained symmetry in the transverse plane of the body. This will allow the rover to flip and continue walking. So if the rover could flip without losing its functionality, the need for the rover to avoid obstacles decreases. This will increase the simplicity and flexibility of the rover. Hence, a flexible solar panel and symmetric body will bring back this benefit.

If a more advanced communication module is installed on the rover during testing days, the team could simulate lunar operations. During the test mission in this thesis, many things were discovered that could be improved. However, while deploying a rover on the moon, the teams need to be sure that the rover will operate as expected and that communication is possible and efficient. The only way to make this happen is to apply more testing phases.

Another interesting recommendation for OPAL is making use of more than just 3D information. Sometimes the information in the picture could enhance the performance. The obstacle cluster sometimes still contains some ground points. This resulted in incorrect bounding boxes and incorrect obstacle distances. There are multiple ways to tackle this problem. The first and most obvious way is to fix the ground plane detection. The inclusion of a method that could use other information present in the image like edges and pixel intensity could help. A known approach in many computer vision algorithms uses so called superpixels, which segments groups of pixels with the same image characteristics. Superpixel segmentation algorithms could improve this distinction between the rocks and the ground. Another approach is to exclude the points close to the ground plane from the distance determination. The lower points, especially the points lower than 30 mm, will not form a threat since they are 'walkable'. It is likely that when these are deducted from the height determination, the ground points will have less influence. Lastly, the current trend within the computer vision research field is to make use of machine learning, which is also making more use of both 3D and image information.

Finally, it may be concluded that development of OPAL is far from finished and with new studies to this obstacle detection system more discussion points and recommendations will arise.

A

Rover Market

In this appendix an overview of the market of planetary rovers is provided. All the rovers are listed in Table A.1.

Rover Project Name	Parent Mission	Year	Mass (kg)	Dimensions (w x l x h)(m)	Distance Traveled(km)	Destination
Lunokhod 1	Luna 17	1970	756	2.2 x 2.2 x 1.5	10.5	Moon (Mare Imbrium)
Lunokhod 2	Luna 21	1973	840	1.7 x 1.6 x 1.35	39	Moon (Le Monnier crater)
Prop-M	Mars 2/3	1971	4.5	0.25 x 0.22 x 0.04	0	Mars
Sojourner	Mars Pathfinder	1997	11.5	0.65 x 0.48 x 0.3	0.1	Mars
Spirit/Opportunity	MER	2003	174	1.5 x 2.3 x 1.6	45.16 + 7.7	Mars
Curiosity	Mars Science Laboratory	2011	899	3 x 2.7 x 2.2	22.97	Mars
Yutu	Chang'e 3	2013	140	1.5 x 1.0 x 1.0	0.115	Moon (Mare Imbrium)
Yutu 2	Chang'e 4	2018	140	1.5 × 1.0 × 1.0	0.5	Moon (Far Side)
Pragyan rover	Chandrayaan-2	2019	27	0.9 x 0.75 x 0.85	0	Moon (South Pole)
Perseverance	Mars 2020	2020	1025	3 x 2.7 x 2.2	-	Mars
Asagumo	mission one	2021	1.3	0.01 x 0.01 x 0.01	-	Moon (Lacus Mortis)
Viper	CLPS	2022	430	1.5 x 1.5 x 2.5	-	Moon (South Pole)

Rover Project Name	Navigation Sensors	Autonomy	Primary Mission Objectives	References
Lunokhod 1	3 cameras	Remote	Collect data of the composition of the regolith	[73] [76] [94]
Lunokhod 2	3 cameras	Remote	Examine ambient light levels and measure local magnetic fields	[54] [76]
Prop-M	2 indicator rods	Autonomous	Measure rocks and soil	[72] [94]
Sojourner	stereo camera with structure light	Semi-Autonomous	Demonstrate that small rovers can actually operate on Mars	[71] [44]
Spirit/Opportunity	2 stereo camera + multiple hazcams	Autonomous	Perform multiple studies of the mars surface	[64] [6] [76] [94]
Curiosity	2 stereo camera + multiple hazcams	Autonomous	Search for possiblity of life on mars	[53] [94]
Yutu	stereo camera + multiple hazcams	Autonomous	Inspect the soil and the structure of the lunar crust	[56] [76] [15]
Yutu 2	stereo camera + multiple hazcams	Autonomous	Measuring radiation, detect sub-surface water.	[56] [15]
Pragyan rover	stereo camera with structure light	Autonomous	Surviving one lunar day, perform on-site analyses	[43]
Perseverance	2 stereo camera + multiple hazcams	Autonomous	Search for life and sample return	[74]
Asagumo	lidar	Autonomous	Demonstrate technology	[98]
Viper	cameras	Remote	Analyse water ice on the surface of the moon	[14]

Table A.1: Planetary Rover Market Overview

B

Test Details Decos

This appendix is describing more details about the conducted tests at the Decos Mars Yard. What is tested on the 21st and 23rd of July is elaborated on in this appendix. First, a detailed list of the test equipment is listed. Hereafter, the test-setup is explained and which rocks are used during testing. At last, the observations about the used positioning system are stated.

All test equipment and the extra test tools are listed in Table B.1. Since the test location at Decos had no easily accessible power source, a remote power source needs to be arranged so the test equipment can operate. The rest of the equipment mainly consists of the test models, their accessories and the positioning system. The

Table B.1: Equipment List

Quatity	Item
1	Test Lunar Zebro
1	Reserve Terrestial Zebro
2	433 MHz Remote Controls
3	3S/4S Lipo Batteries
2	12V Lead Acid Accu
1	12V Power Source + LiPo Battery Charger
1	12V to 220V DC-AC Converter
4	Foldable Stands
4	Pinxact Anchors
1	Pinxact Portable Tag
1	Laser Distance Measurement Unit
4	10m Power Cords
2	Power Plug Boxes
1	5V Portable Router
-	Laptops
-	Toolboxes
-	Chairs/Table/Tent
-	Measuring Tape/Tools
-	Calibration Boards

position of the rover was recorded by the localisation system of PinXact. This localisation system uses four anchors, which are sending an Ultra-WideBand (UWB) radio-frequency. With this UWB radiofrequency, the system seeks to measure the difference in time of arrival. The difference in arrival time is proportional to the tag's distance concerning the anchors inside the grid. This test setup is shown in Figure B.7. The four anchors were placed in a square of 10.7 m and at a height of 1.8 m. One tag was inside the rover, but a portable tag measured the position of the rocks with the localisation system. The portable tag is depicted in Figure B.8

After setting up the testing grid, some rocks of the simulated Martian surface were moved inside the grid.

The different rocks which were used during testing are depicted in Figures B.1 trough B.6. A list of the rocks, their location and characteristics, is provided in Table B.2. The rocks are also shown in figure B.7.

Unfortunately, the accuracy of the localisation data produced by this PinXact Ultra Wide Band DecaWave

Table B.2: Rock Characteristic List (units in cm)

Nr.	X	Y	Z	Widht(x)	Length(y)	Height(z)	Obstacle Description
1	281	501	25	28	35	31	Big brown gray rock
2	550	516	26	50	25	20	Long white rock
3	664	651	131	16	19	12	Small gray rock (right)
4	612	648	81	16	23	13	Small gray rock (left)
5	312	718	93	13	12	5	Group of tiny white/gray rocks
6	382	822	78	18	4	4	Walkable long rock
7	495	867	55	30	28	29	Group of small brown rocks

Figure B.1: Obstacle One

Figure B.2: Obstacle Two

Figure B.3: Obstacle Three and Four

Figure B.4: Obstacle Five

Figure B.5: Obstacle Six

Figure B.6: Obstacle Seven

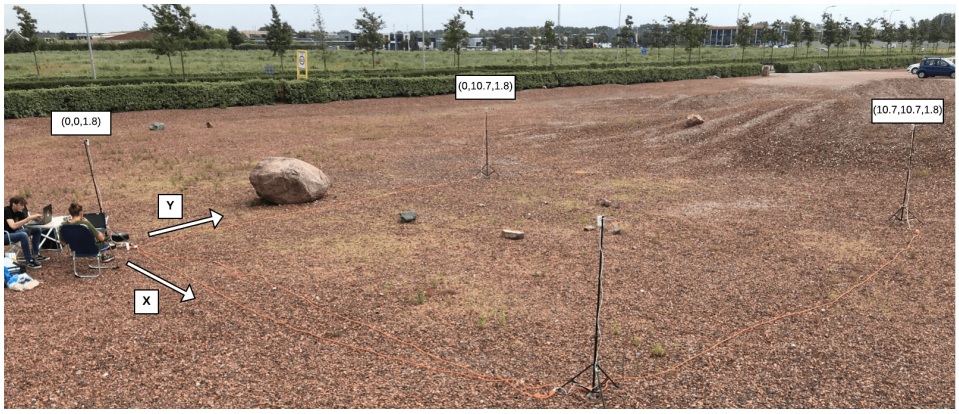


Figure B.7: Test Setup with the Pinxact Anchors on every corner

Figure B.8: Tag Connected to a Raspberry Pi and Powerbank

system turned out to be accurate to around 10 cm. The stereo vision system has an error of 4.8 cm at the detection-distance. Nevertheless, having localisation data with a more significant error as the theoretical error of the stereo camera system is unsuitable for verification.

Fortunately, during the test day at Decos tape measurements were conducted. This additional measurement method is finally used in the upcoming case validation. In Figure B.9, the localisation data is visualised by the red line. As seen, this data is very noisy. When smoothed by a simple smoothing filter, the localisation data seems closer to reality, as visualised by the blue line. However, it can be seen that the location of the rock is uncertain. In an ideal situation, the added circles need to cross each other at one point, such as where the ellipse is drawn. The ellipse represents the rock in this case. Nevertheless, since there are many circles not close to intersecting, this shows that the data is unreliable. Therefore, only the tape measurements are used.

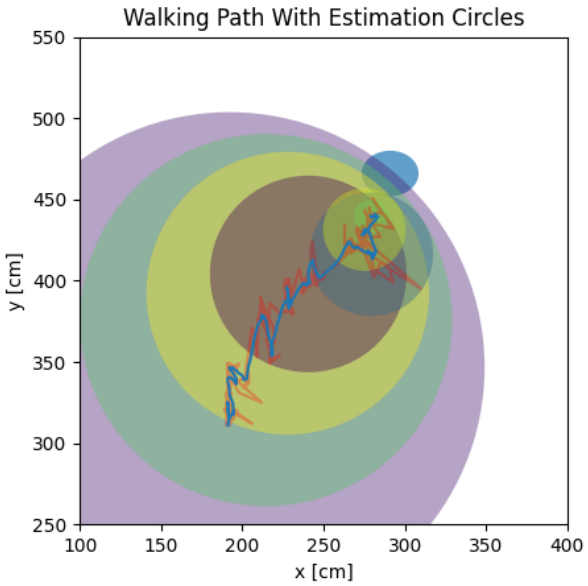


Figure B.9: Walking path estimated with the PinXact localisation system

C

Matlab Calibration Results

In this appendix, all the fourteen Matlab calibration results of the Decos tests are listed. The calibrations were spread over the length of a testing day. In the first graph, the spread of every chequerboard position is shown. The second graph shows the re-projection error of these positions with the calculated camera model parameters, the overall mean error of each calibration is also provided. This error does not exceed 0.16 pixels or higher.

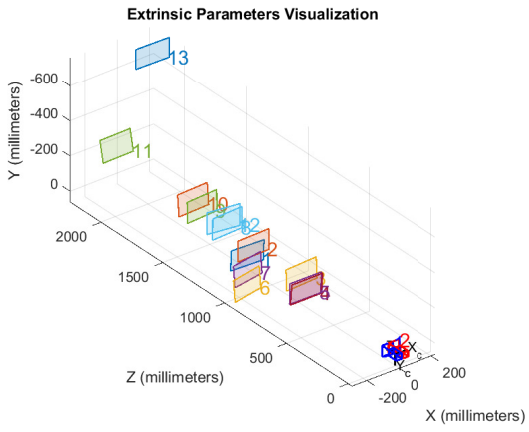


Figure C.1: Calibration 1: Extrinsic Parameters Visualization

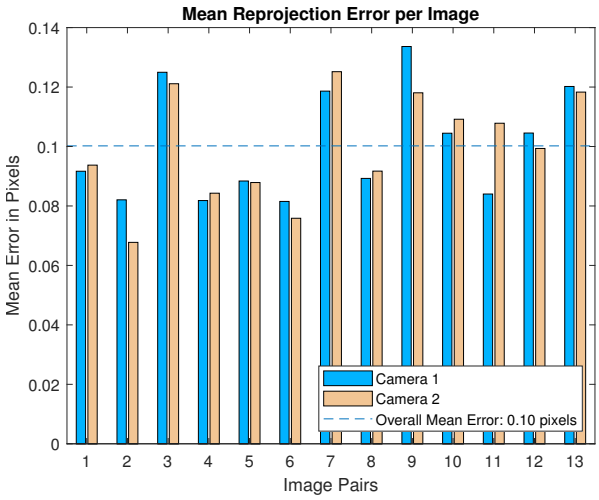


Figure C.2: Calibration 1: Mean Reprojection Error per Image

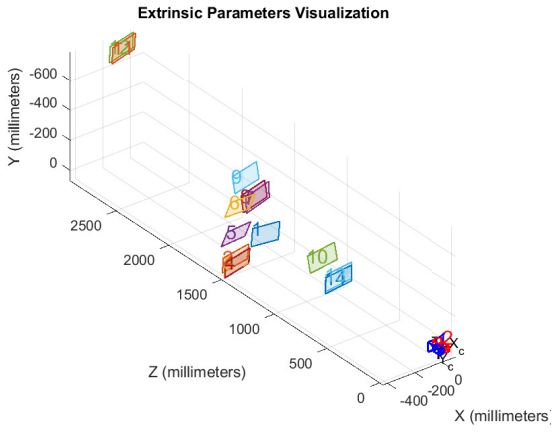


Figure C.3: Calibration 2: Extrinsic Parameters Visualization

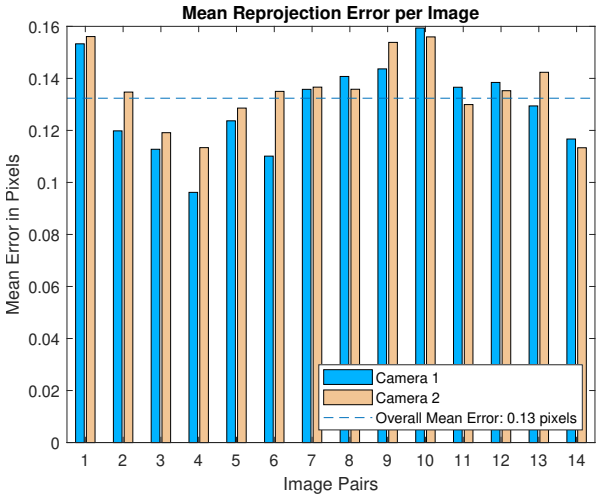


Figure C.4: Calibration 2: Mean Reprojection Error per Image

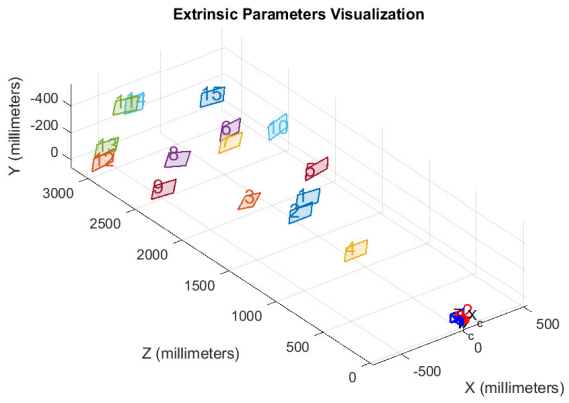


Figure C.5: Calibration 3: Extrinsic Parameters Visualization

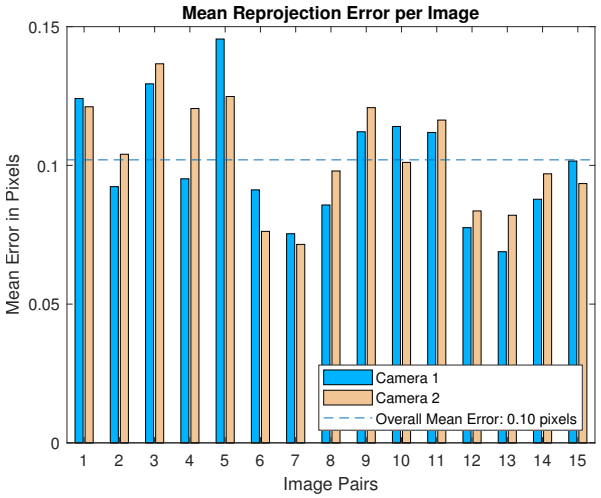


Figure C.6: Calibration 3: Mean Reprojection Error per Image

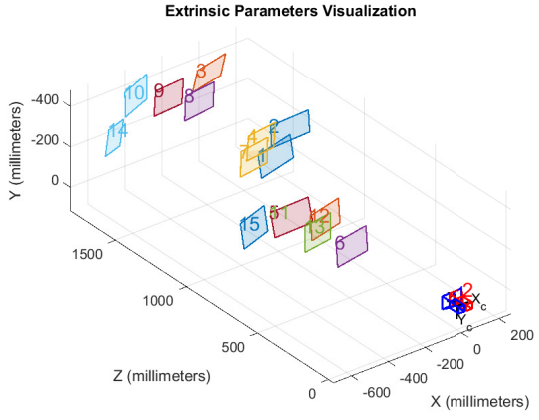


Figure C.7: Calibration 4: Extrinsic Parameters Visualization

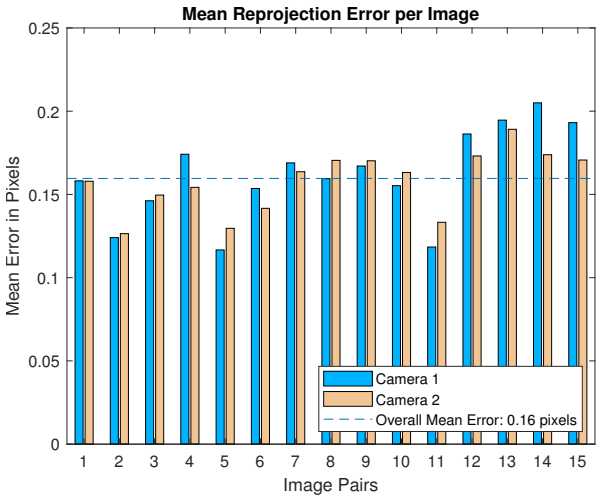


Figure C.8: Calibration 4: Mean Reprojection Error per Image

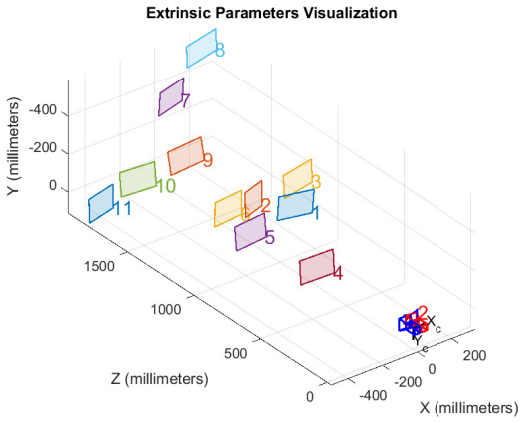


Figure C.9: Calibration 5: Extrinsic Parameters Visualization

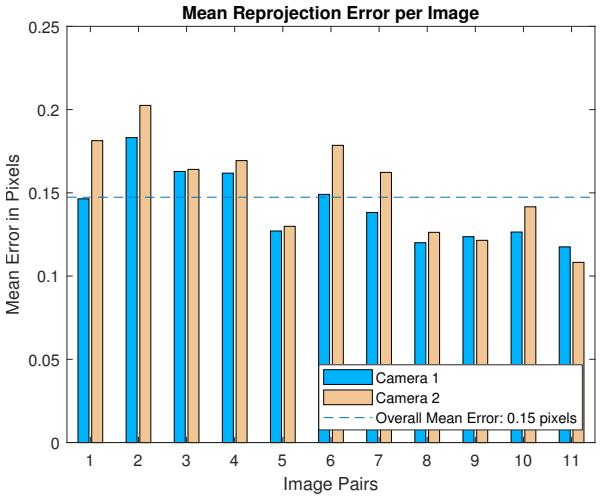


Figure C.10: Calibration 5: Mean Reprojection Error per Image

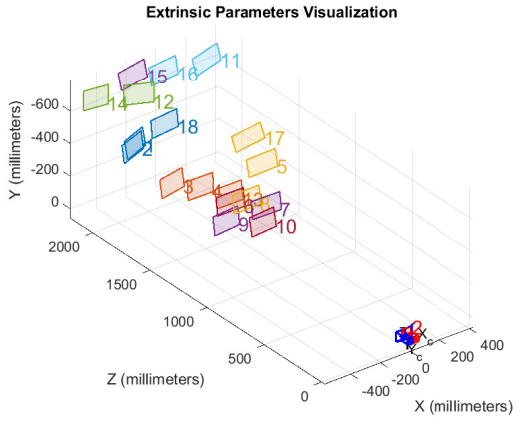


Figure C.11: Calibration 6: Extrinsic Parameters Visualization

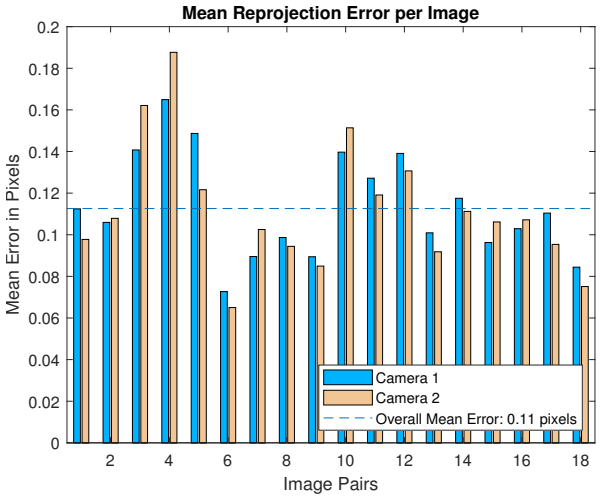


Figure C.12: Calibration 6: Mean Reprojection Error per Image

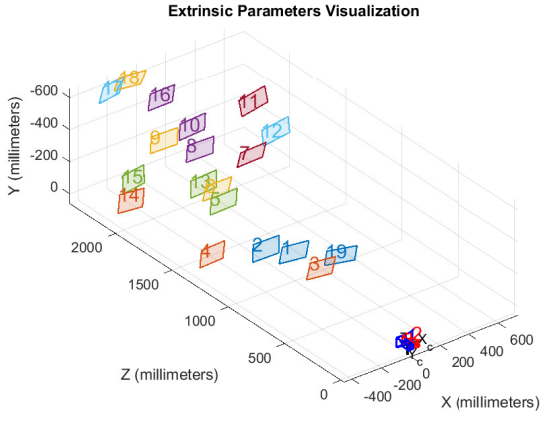


Figure C.13: Calibration 7: Extrinsic Parameters Visualization

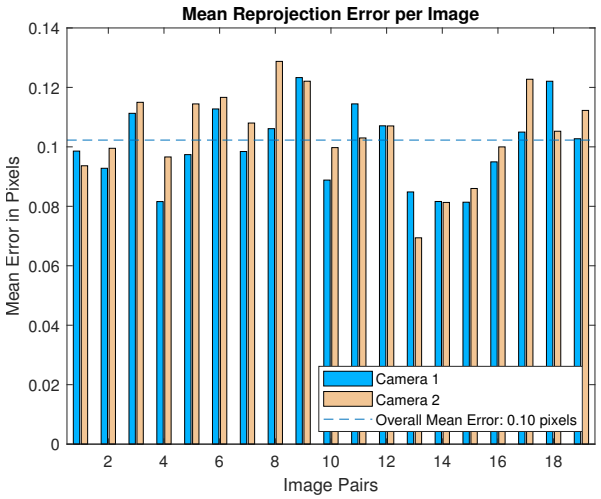


Figure C.14: Calibration 7: Mean Reprojection Error per Image

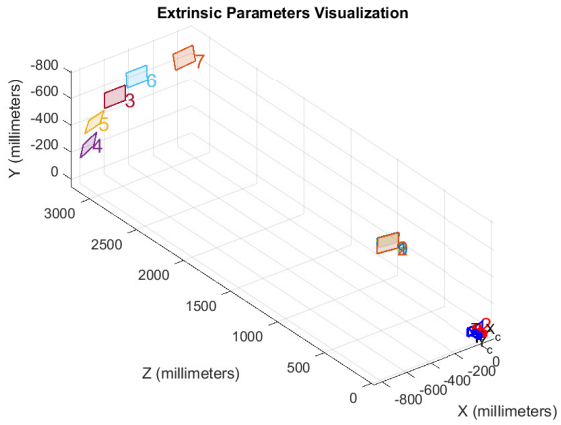


Figure C.15: Calibration 8: Extrinsic Parameters Visualization

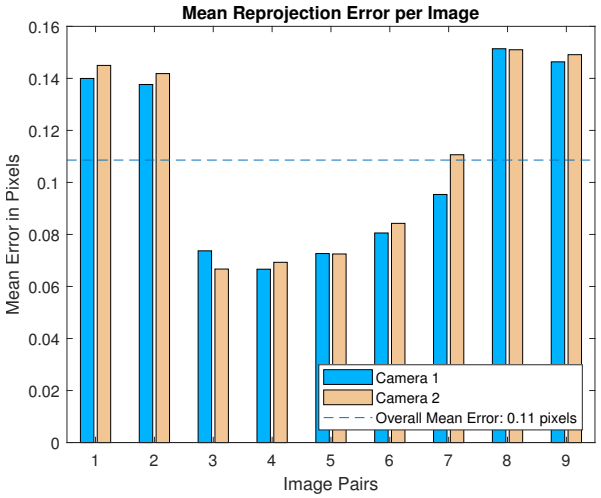


Figure C.16: Calibration 8: Mean Reprojection Error per Image

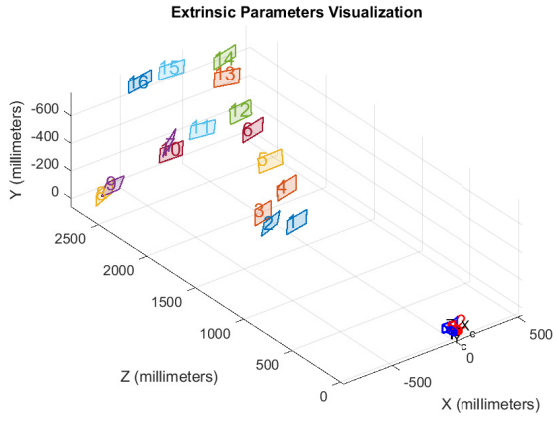


Figure C.17: Calibration 9: Extrinsic Parameters Visualization

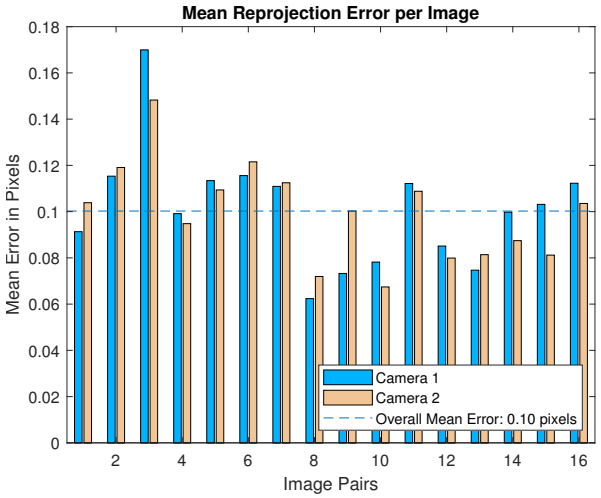


Figure C.18: Calibration 9: Mean Reprojection Error per Image

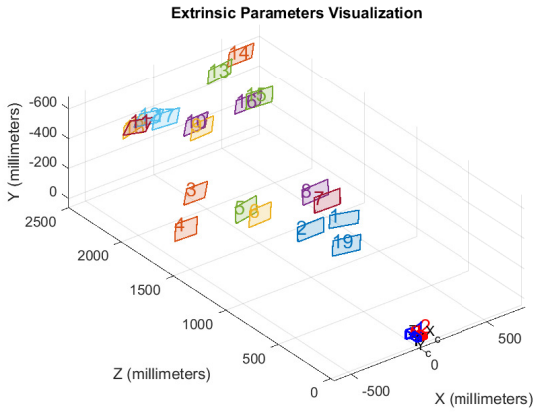


Figure C.19: Calibration 10: Extrinsic Parameters Visualization

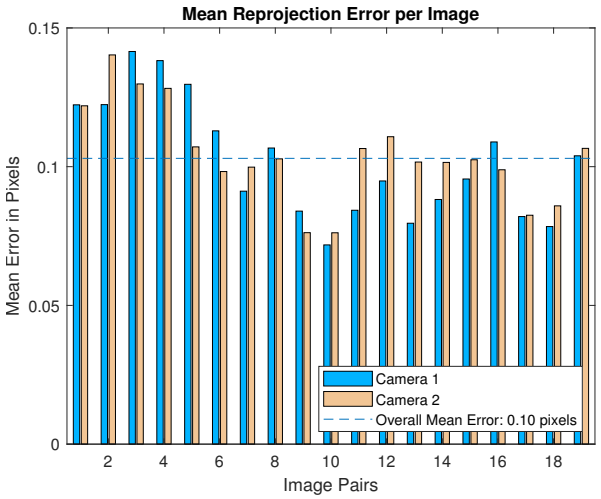


Figure C.20: Calibration 10: Mean Reprojection Error per Image

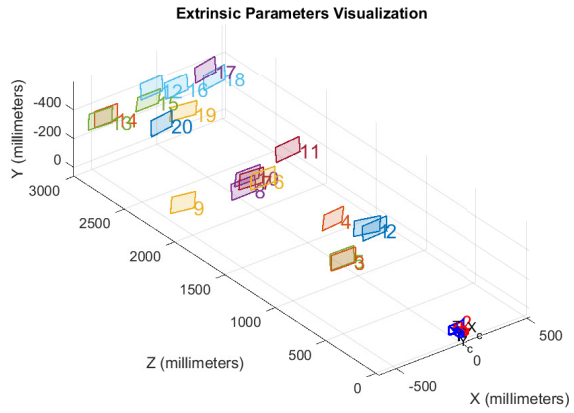


Figure C.21: Calibration 11: Extrinsic Parameters Visualization

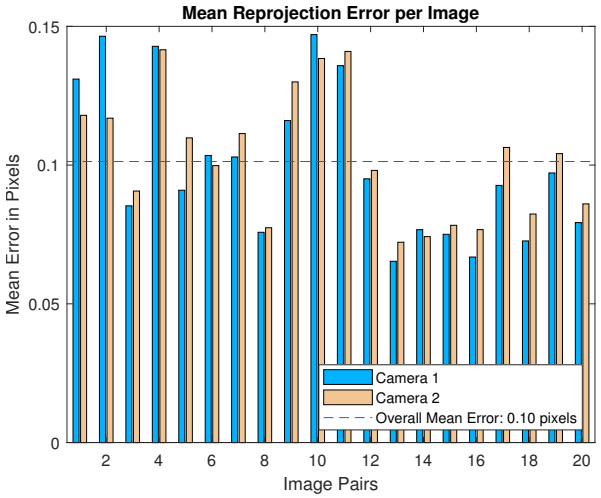


Figure C.22: Calibration 11: Mean Reprojection Error per Image

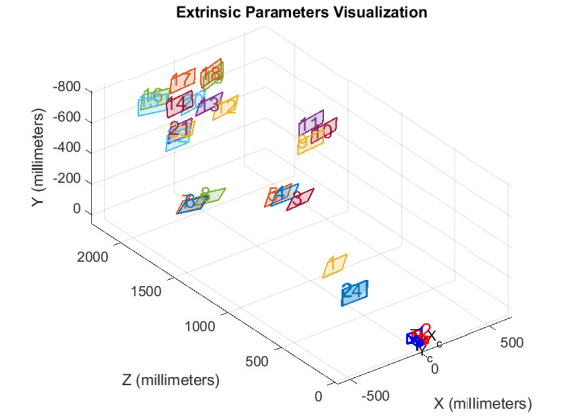


Figure C.23: Calibration 12: Extrinsic Parameters Visualization

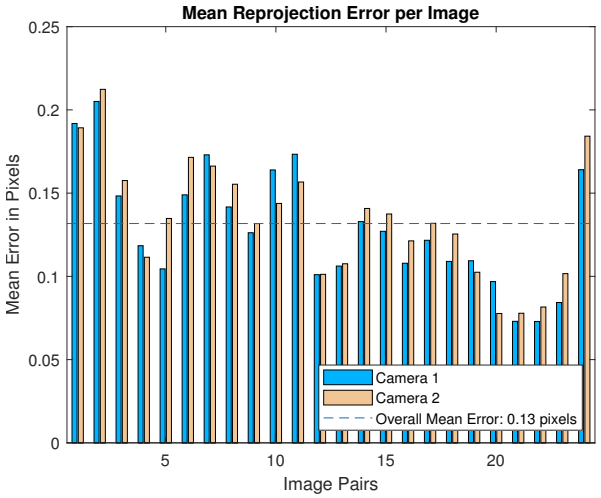


Figure C.24: Calibration 12: Mean Reprojection Error per Image

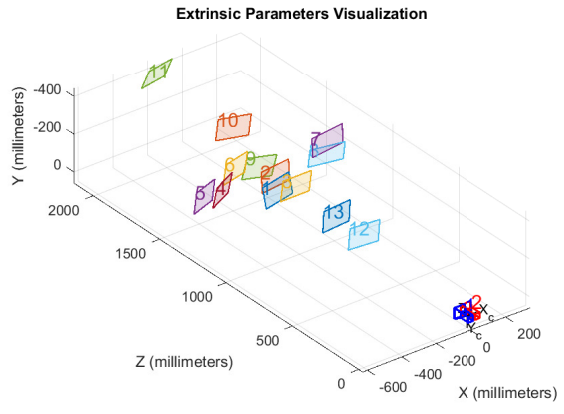


Figure C.25: Calibration 13: Extrinsic Parameters Visualization

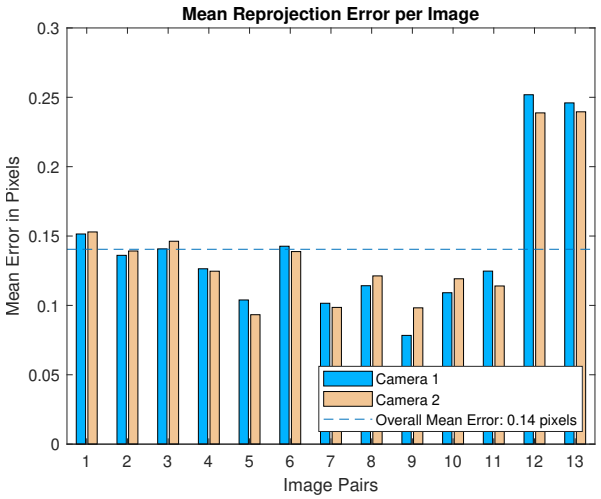


Figure C.26: Calibration 13: Mean Reprojection Error per Image

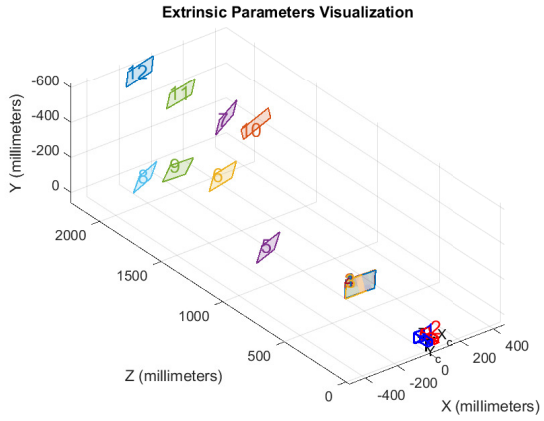


Figure C.27: Calibration 14: Extrinsic Parameters Visualization

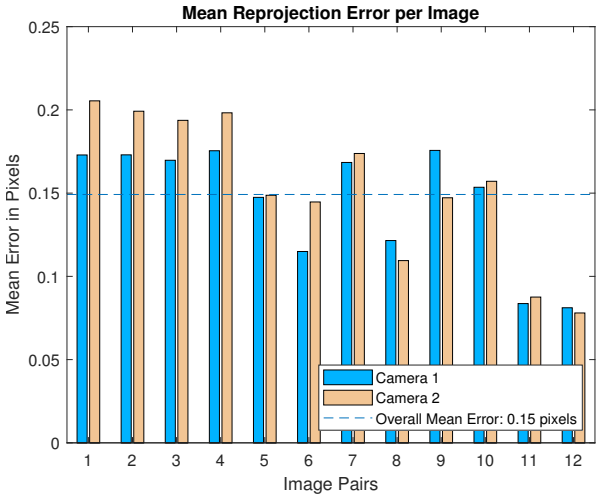


Figure C.28: Calibration 14: Mean Reprojection Error per Image

D

PCB Design

The new Carrier Board Design KiCAD drawings are shown in this Appendix. The 4 layer lay-up of the PCB is also shown.

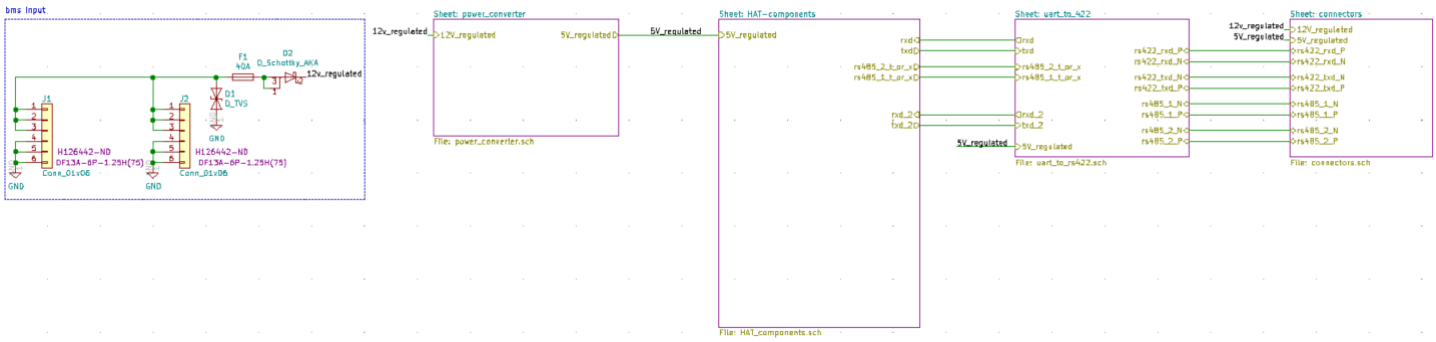


Figure D.1: KiCAD PCB Design Overview Sheet

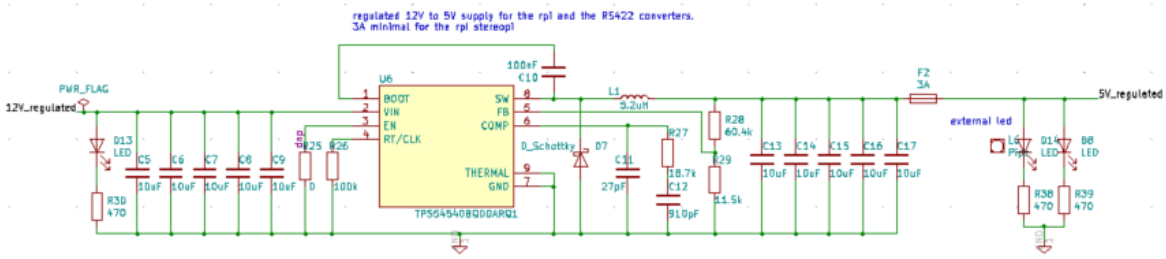


Figure D.2: KiCAD PCB Design Power-converter Sheet

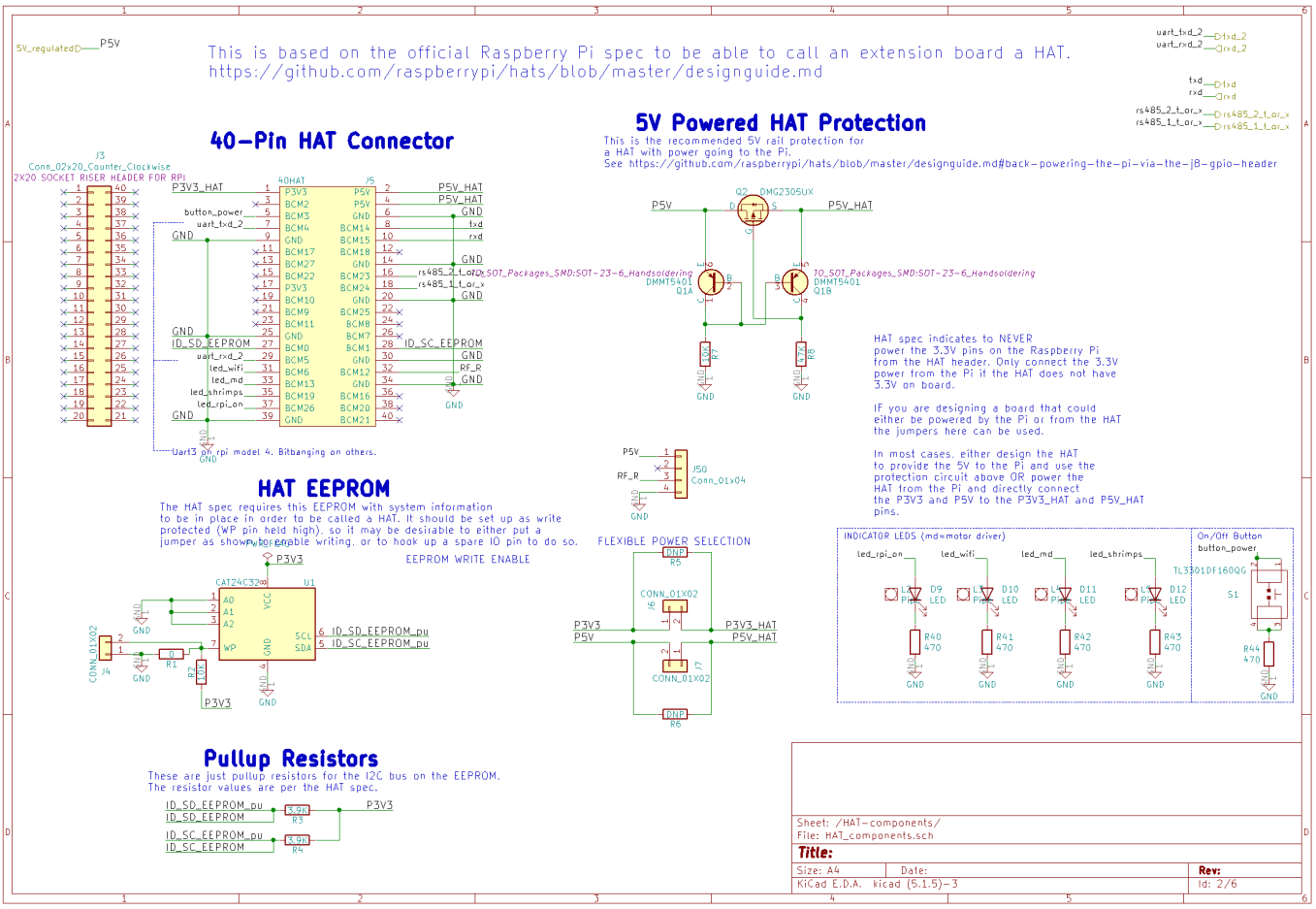


Figure D.3: KiCAD PCB Design Raspberry Pi Header Components Sheet

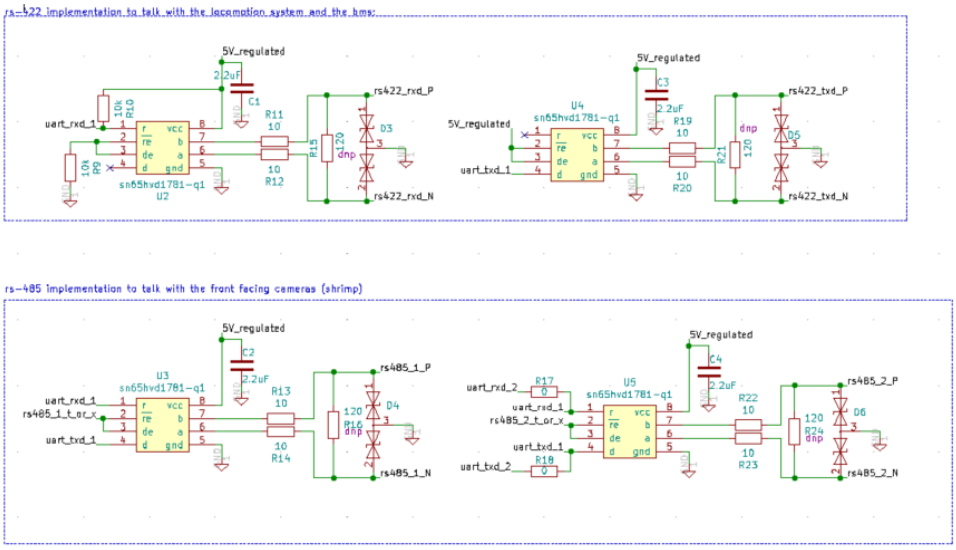


Figure D.4: KiCAD PCB Design Communication Distribution Sheet

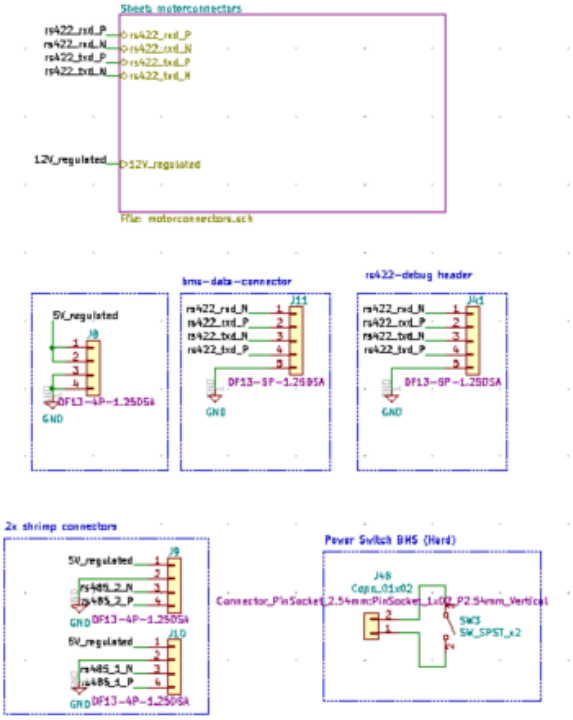


Figure D.5: KiCAD PCB Design Connectors Sheet

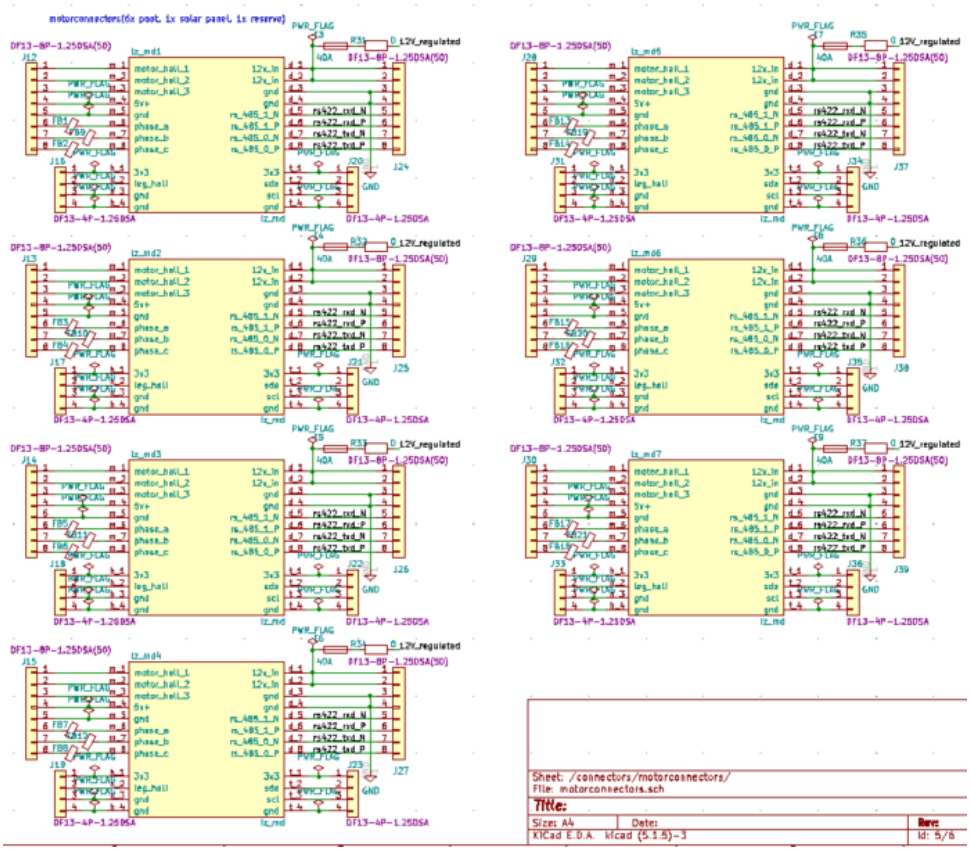


Figure D.6: KiCAD PCB Design Motor-connectors Sheet

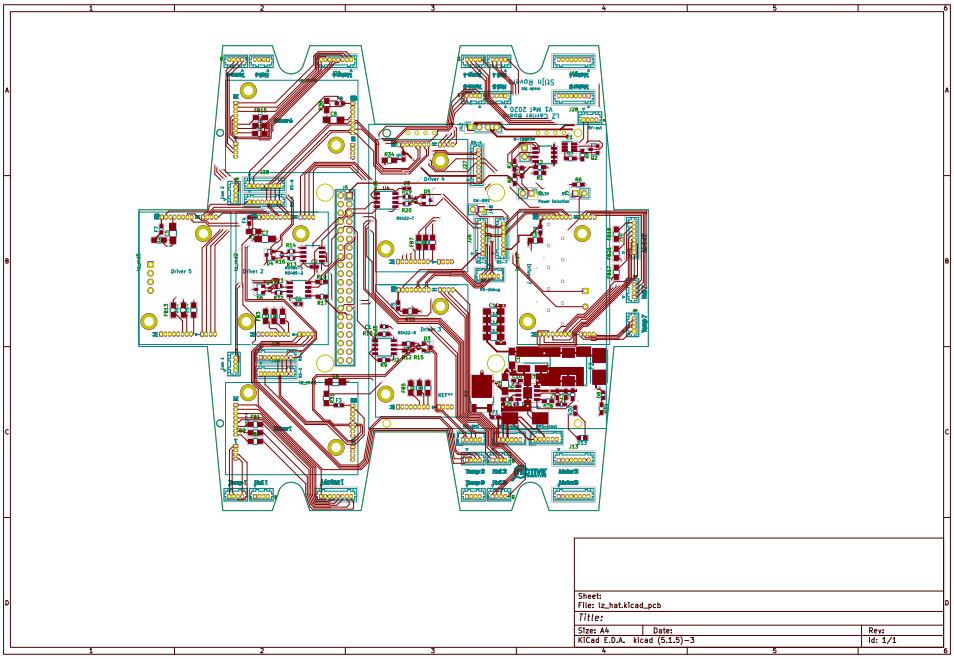


Figure D.7: KiCAD PCB Design Top-Layer Layout

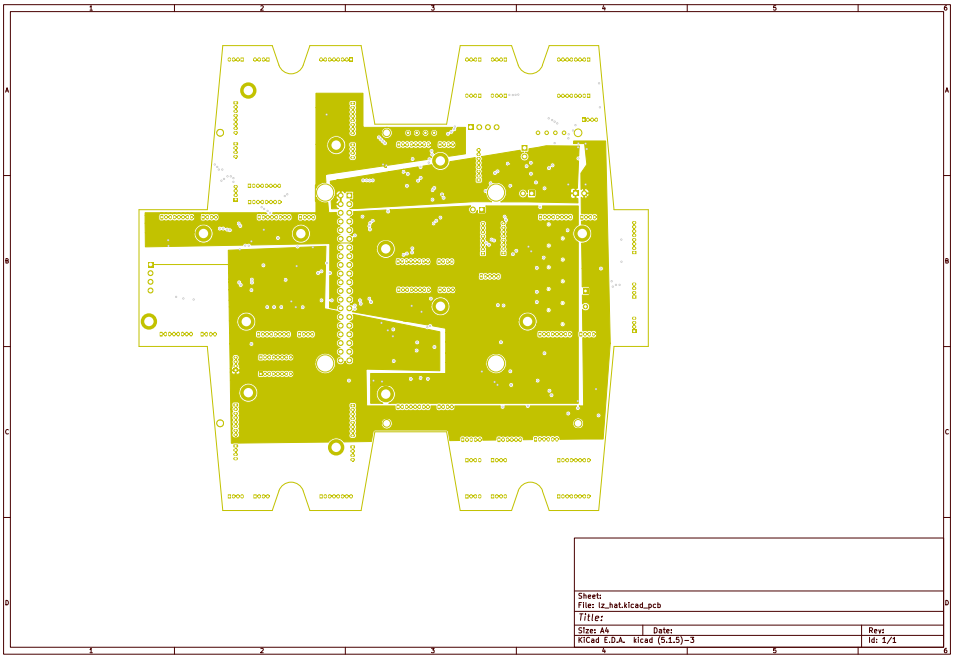


Figure D.8: KiCAD PCB Design First Middle-Layer Layout

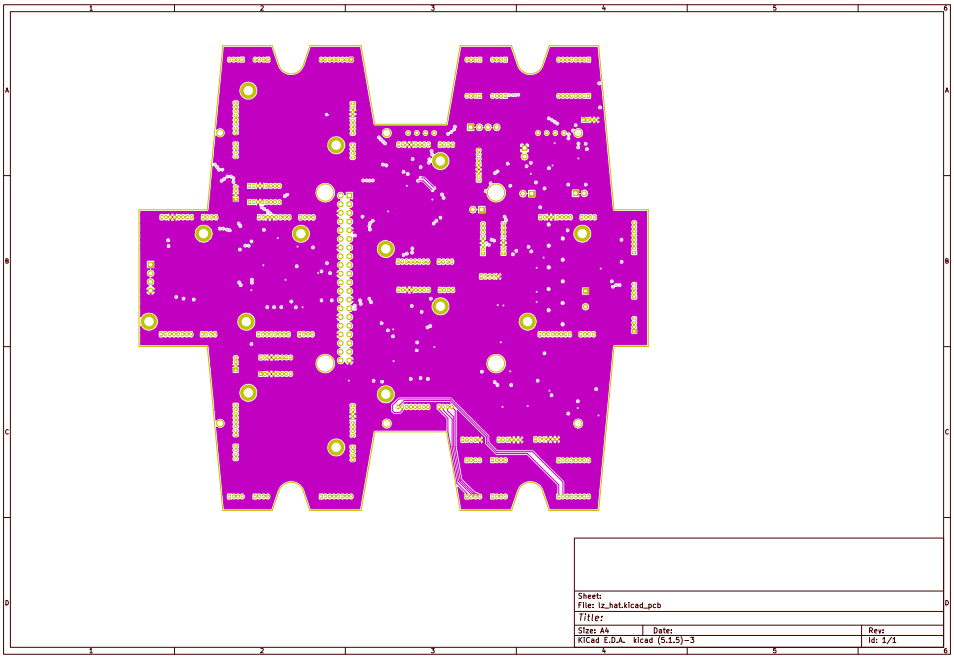


Figure D.9: KiCAD PCB Design Second Middle-Layer Layout

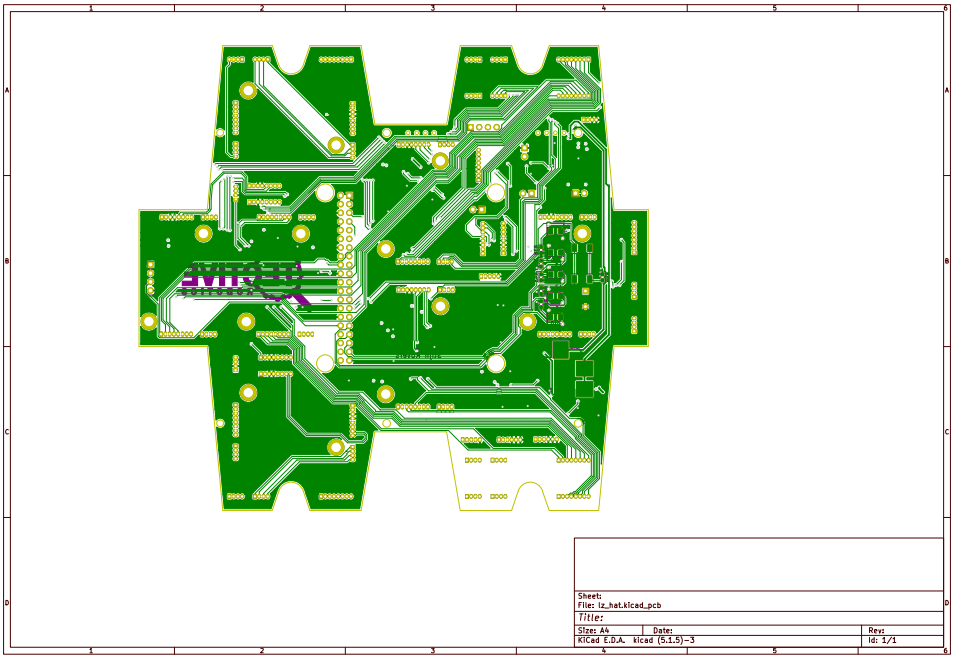


Figure D.10: KiCAD PCB Design Bottom-Layer Layout

E

Stereo Matching Parameters Optimizing GUI

In this appendix some screen shots are included of the working GUI for optimizing stereo matching parameters. This GUI, running on Linux, was created using QT and is based on a existing matching parameters tuner made for the Block Matching algorithm of OpenCV [84].

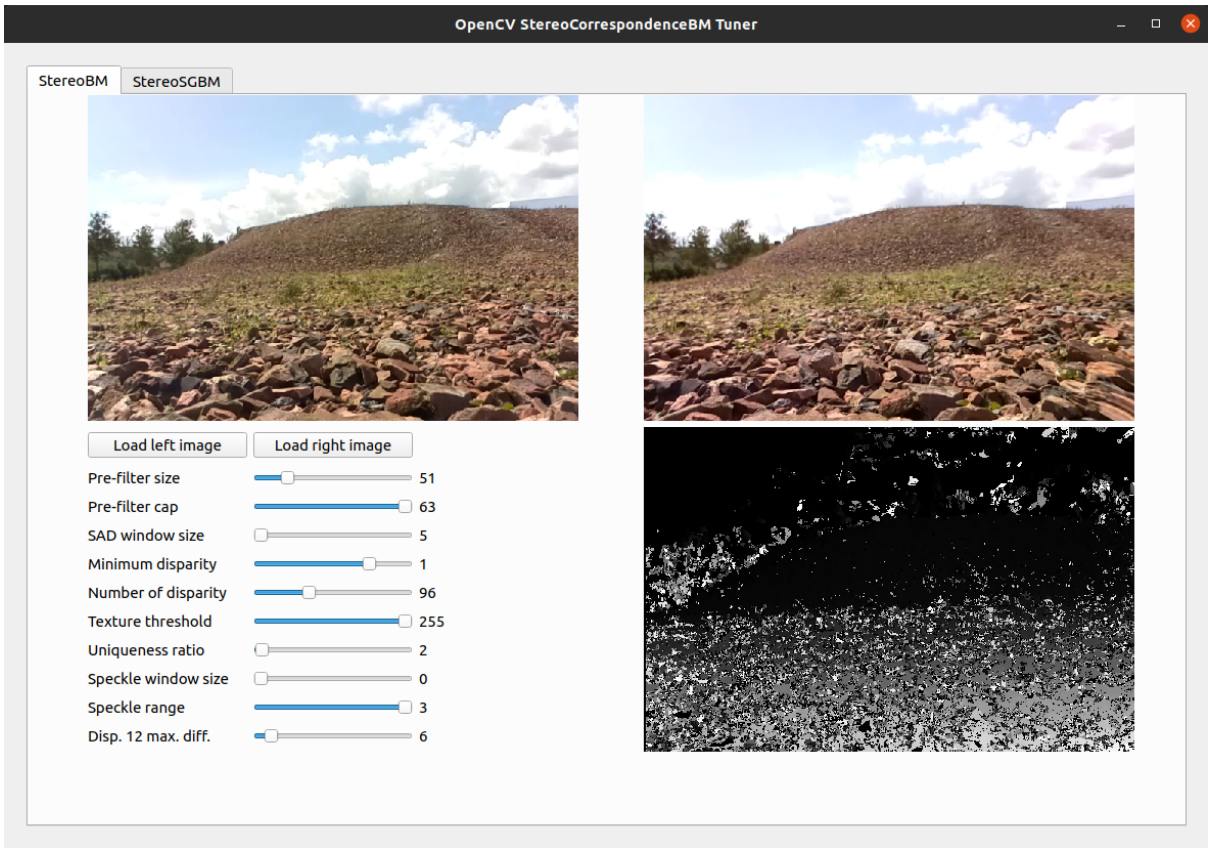


Figure E.1: GUI Block Matching Implementation

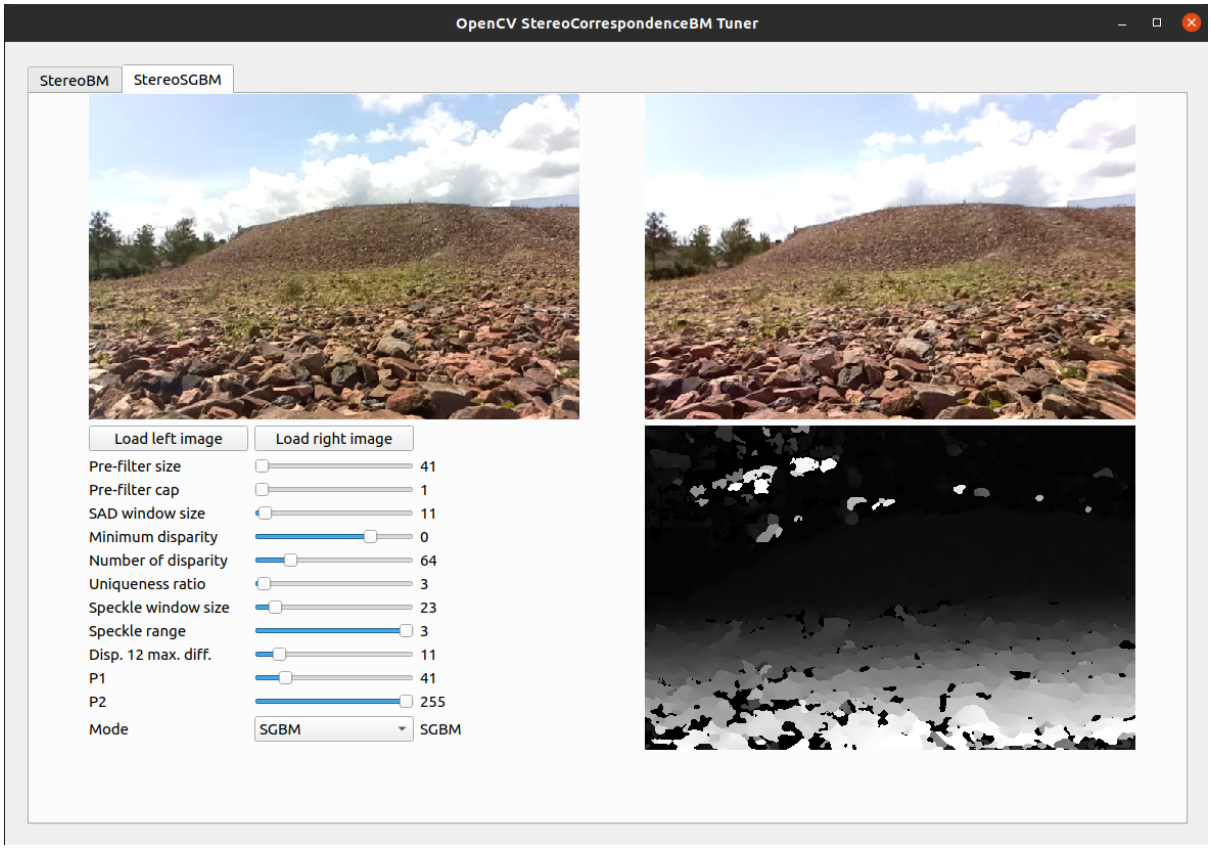


Figure E.2: GUI Semi Global Block Matching Implementation

Bibliography

[1] Arthur Admiraal, Nick Belzer, Bas van Vliet, Richard van Groenendijk, Jasper Jonk, and Stephan Rutten. SHRIMP - Payload design report. Technical Report April, Delft, University of Technology, 2018.

[2] João Aguiar. Underwater Stereoscopic Vision with Convergent Cameras. Technical report, Faculdade de Engeharia da Universidade do Porto, Porto, 2015.

[3] Adnan Ansar, Andres Castano, and Larry Matthies. Enhanced Real-time Stereo Using Bilateral Filtering. *Jet Propulsion*, pages 1–8, 2004.

[4] C. Banz, P. Pirsch, and H. Blume. Evaluation of penalty functions for semi-global matching cost aggregation. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXIX-B3:1–6, jul 2012. ISSN 1682-1750. doi: 10.5194/isprsarchives-xxxix-b3-1-2012.

[5] M. J. Bentum, M. K. Verma, R. T. Rajan, A. J. Boonstra, C. J.M. Verhoeven, E. K.A. Gill, A. J. van der Veen, H. Falcke, M. Klein Wolt, B. Monna, S. Engelen, J. Rotteveel, and L. I. Gurvits. A roadmap towards a space-based radio telescope for ultra-low frequency radio astronomy. *Advances in Space Research*, 65 (2):856–867, 2019. ISSN 18791948. doi: 10.1016/j.asr.2019.09.007. URL https://doi.org/10.1016/j.asr.2019.09.007.

[6] J.J. Biesiadecki and M.W. Maimone. The Mars Exploration Rover Surface Mobility Flight Software: Driving Ambition. *IEEE Aerospace Conference*, pages 1–15, 2006. doi: 10.1109/aero.2006.1655723.

[7] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S. Davis. Soft-NMS - Improving Object Detection with One Line of Code. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-October:5562–5570, 2017. ISSN 15505499. doi: 10.1109/ICCV.2017.593.

[8] G. Boothroyd and L. Alting. Design for Assembly and Disassembly. *CIRP Annals - Manufacturing Technology*, 41(2):625–636, 1992. ISSN 17260604. doi: 10.1016/S0007-8506(07)63249-1.

[9] Boston Dynamics. Rhex, 2012. URL https://www.bostondynamics.com/rhex.

[10] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[11] John Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986. ISSN 01628828. doi: 10.1109/TPAMI.1986.4767851.

[12] Fengping Cao and Rongben Wang. Study on stereo matching algorithm for lunar rover based on multi-feature. In *CICC-ITOE 2010 - 2010 International Conference on Innovative Computing and Communication, 2010 Asia-Pacific Conference on Information Technology and Ocean Engineering*, pages 209–212. IEEE Computer Society, jan 2010. ISBN 9780769539423. doi: 10.1109/CICC-ITOE.2010.60.

[13] Kenneth Chang. Why Everyone Wants to Go Back to the Moon, 2019. URL https://nyti.ms/2xIUZDh.

[14] Rick Chen. VIPER, 2020. URL http://www.nasa.gov/viper.

[15] China National Space Administration. Yutu Rover, 2013. URL http://moon.bao.ac.cn/.

[16] Cooper, R. What is Stereo Vision?, 1995. URL https://www.vision3d.com/stereo.html.

[17] E. Dall'Asta and R. Roncella. A comparison of semiglobal and local dense matching algorithms for surface reconstruction. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, 40(5):187–194, 2014. ISSN 16821750. doi: 10.5194/isprsarchives-XL-5-187-2014.

[18] Daniel Scharstein, Richard Szeliski, and Heiko Hirschmüller. Middlebury Stereo Vision Page, 2002. URL https://vision.middlebury.edu/stereo/.

[19] Jurriaan de Groot. *Swarm Behaviour For The Zebro Robot*. PhD thesis, Delft, University of Technology, 2017.

[20] C. De Wagter, S. Tijmons, B. D.W. Remes, and G. C.H.E. De Croon. Autonomous flight of a 20-gram Flapping Wing MAV with a 4-gram onboard stereo vision system. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4982–4987, 2014. ISSN 10504729. doi: 10.1109/ICRA.2014.6907589.

[21] J. L. Deneubourg, S. Aron, S. Goss, and J. M. Pasteels. The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, 3(2):159–168, 1990. ISSN 08927553. doi: 10.1007/BF01417909.

[22] Kaichang Di, Zongyu Yue, Zhaoqin Liu, and Shuliang Wang. Automated rock detection and shape analysis from mars rover imagery and 3D point cloud data. *Journal of Earth Science*, 24(1):125–135, 2013. ISSN 1674487X. doi: 10.1007/s12583-013-0316-3.

[23] Gijs Dubbelman. Vision based Obstacle Detection for both Day and Night Conditions. Technical report, University of Amsterdam, Amsterdam, 2006.

[24] Heather Dunlop. *Automatic Rock Detection and Classification in Natural Scenes*. PhD thesis, Carnegie Mellon University, 2006.

[25] Nick Van Endhoven. How to implement the Brain of a Moon Rover. Technical report, Delft, University of Technology, 2020.

[26] Ronald Ensing. 3D Robot Vision (ME47030) Lecture 4 : Stereo Matching, 2019.

[27] Martin Ester, Hans-Peter Kriegel, Sander Jorg, and Xiaowei Xu. A Density-Based Clustering Algorithms for Discovering Clusters. *KDD-96 Proceedings*, 96(34):226–231, 1996. ISSN 09758887. doi: 10.1016/B978-044452701-1.00067-3.

[28] Mark Everingham, Luc Van Gool, Christopher K I Williams, John Winn, Andrew Zisserman, M Everingham, L KU Van Gool Leuven, Belgium CKI Williams, J Winn, and A Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. *Int J Comput Vis*, 88:303–338, 2010. doi: 10.1007/s11263-009-0275-4. URL `http://www.flickr.com/`.

[29] Fabian Pedregosa. GitHub - memory-profiler: Monitor Memory usage of Python code, 2014. URL `https://github.com/pythonprofilers/memory_profiler`.

[30] Zeev Farbman, Raanan Fattal, Dani Lischinski, and Richard Szeliski. Edge-preserving decompositions for multi-scale tone and detail manipulation. In *ACM Transactions on Graphics (TOG)*, volume 27, page 67. ACM, 2008.

[31] Colleen M. Farrelly. Comparison of 6 K-Means Clustering Algorithms under Differing Data Conditions. *BMC Bioinformatics*, pages 1–56, 2014. URL `http://weekly.cnbnews.com/news/article.html?no=124000`.

[32] C R Fonville. The Exploring DelFly How to increase the indoor explored area of the DelFly Explorer by means of computationally efficient routing decisions? Technical report, Delft University of Technology, 2016.

[33] Raspberry Pi Foundation. About Us, 2012. URL `https://www.raspberrypi.org/about/`.

[34] Henri P. Gavin. The Levenburg-Marqurdt Algorithm For Nonlinear Least Squares Curve-Fitting Problems. *Department of Civil and Environmental Engineering, Duke University*, pages 1–19, 2019. URL `http://people.duke.edu/{~}hpgavin/ce281/lm.pdf`.

[35] Dariu M Gavrila. 3D Robot Vision (ME47030) Lecture 3 : Stereo Geometry, 2019.

[36] William Gerstenmaier and Jason Crusan. Cislunar and Gateway Overview, 2018.

[37] Steven B Goldberg, Mark W Maimone, and Lany Matthies. Stereo Vision and Rover Navigation Software for Planetary Exploration. *IEEE Aerospace Conference Proceedings*, 5(March):2025–2036, 2002.

[38] Google AI. Open Images - Object Detection Track | Kaggle, 2019. URL https://www.kaggle.com/c/google-ai-open-images-object-detection-track.

[39] Brian Harvey. *Soviet and Russian lunar exploration*, volume 45. Springer, 2013. ISBN 9780387218960. doi: 10.5860/choice.45-0842.

[40] Allam Shehata Hassanein, Sherien Mohammad, Mohamed Sameer, and Mohammad Ehab Ragab. A Survey on Hough Transform, Theory, Techniques and Applications. *International Journal of Computer Science Issues*, 2015. URL http://arxiv.org/abs/1502.02160.

[41] G. H. Heiken, D. T. Vaniman, and M. F. Bevan, editors. *Lunar Source Book*. Cambridge University Press, 1991.

[42] Heiko Hirschmüller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, 2008. ISSN 01628828. doi: 10.1109/TPAMI.2007.1166.

[43] Indian Space Research Organisation. GSLV-F10/Chandrayaan-2 Mission, 2019. URL https://www.isro.gov.in/gslv-f10-chandrayaan-2-mission.

[44] J. Matijevic. Mars Pathfinder Microrover - Implementing A Low Cost Planetary Mission Experiment. *Proceedings of the Second IAA International Conference on Low-Cost Planetary Missions*, 1(c):1–30, 1996. ISSN 1098-6596. doi: 10.1.1.50.4823.

[45] Andrew Jones. Chang'e-4 Updates: Yutu-2 Roves into Overtime, Returns More Images, 2019. URL http://www.planetary.org/blogs/guest-blogs/2019/change-4-updates-day-4.html.

[46] S. K. Katiyar and P. V. Arun. Comparative analysis of common edge detection techniques in context of object extraction. *IEEE TGRS*, 50(11b), feb 2014. URL http://arxiv.org/abs/1405.6132.

[47] Joshua Kelcey and Arko Lucieer. Sensor correction of a 6-band multispectral imaging sensor for UAV remote sensing. *Remote Sensing*, 4(5):1462–1493, 2012. ISSN 20724292. doi: 10.3390/rs4051462.

[48] R. Babuska Kersbergen, G.A.D. Lopes, T.J.J. van den Boom, B. De Schutter. Optimal gait switching for legged locomotion. *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'11)*, 19:2729–2734, 2011.

[49] Denis Klimentjew, Andre Stroh, Sascha Jockel, and Jianwei Zhang. Real-time 3D environment perception: An application for small humanoid robots. *2008 IEEE International Conference on Robotics and Biomimetics, ROBIO 2008*, pages 354–359, 2009. doi: 10.1109/ROBIO.2009.4913029.

[50] Kurt Konolige. Small Vision Systems: Hardware and Implementation, 1997. URL http://www.ai.sri.com/$\sim$konolige/svs/disparity.htm.

[51] Julian Kooij. 3D Robot Vision (ME47030) Lecture 2: Single camera models and calibration Image sensing, 2019.

[52] R. Labayrade, D. Aubert, and J.-P. Tarel. Real time obstacle detection in stereovision on non flat road geometry through "v-disparity" representation. *Intelligent Vehicle Symposium, 2002. IEEE*, pages 646–651, 2003. doi: 10.1109/ivs.2002.1188024.

[53] Emily Lakdawalla. *The Design and Engineering of Curiosity*. Springer, 2018. ISBN 9783319681443. doi: 10.1007/978-3-319-68146-7.

[54] Rene Lauper. *Lunar Mission BW1 : Scientific Objectives and Small Satellite Concept*. PhD thesis, Universitat Stuttgart, 2010.

[55] Fei-Fei Li and Professor Fei-Fei Li Stanford. Lecture 4: Finding lines: from detection to model fitting, 2011.

[56] Minglei Li, Zezhou Sun, Shaochuang Liu, Youqing Ma, Hao Ma, Changming Sun, and Yang Jia. Stereo vision technologies for China's lunar rover exploration mission. *International Journal of Robotics and Automation*, 31(2):128–136, 2016. ISSN 08268185. doi: 10.2316/Journal.206.2016.2.206-4440. URL http://www.actapress.com/PaperInfo.aspx?paperId=45347.

[57] Chengwei Liu, Xiubao Sui, Xiaodong Kuang, Yuan Liu, Guohua Gu, and Qian Chen. Adaptive contrast enhancement for infrared images based on the neighborhood conditional histogram. *Remote Sensing*, 11(11), jun 2019. ISSN 20724292. doi: 10.3390/rs11111381.

[58] Charles Loop and Zhengyou Zhang. Computing rectifying homographies for stereo vision. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:125–131, 1999. ISSN 10636919. doi: 10.1109/cvpr.1999.786928.

[59] Alan Lukežič, Tomáš Vojíř, Lukačehovin Luka˘, Lukačehovin Zajc, Jiří Matas, and Matej Kristan. Discriminative Correlation Filter with Channel and Spatial Reliability. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6309–6318, 2017.

[60] Hao Ma, Minglei Li, Changming Sun, Youqing Ma, Yang Jia, Shaochuang Liu, and Zezhou Sun. Stereo Vision Technologies for China's Lunar Rover Exploration Mission. *International Journal of Robotics and Automation*, 31(2), 2016. ISSN 1925-7090. doi: 10.2316/journal.206.2016.2.206-4440. URL http://www.actapress.com/PaperInfo.aspx?paperId=45347.

[61] Roberto Manduchi, Andres Castano, Ashit Talukder, and Larry Matthies. Obstacle detection and terrain classification for autonomous off-road navigation. *Autonomous Robots*, 18(1):81–102, 2005. ISSN 09295593. doi: 10.1023/B:AURO.0000047286.62481.1d.

[62] G. Marirrodriga. Micro-rovers for scientific applications on mars or moon missions. *Preparing for the Future, 7(2), 16–17.*, 1997.

[63] Larry Matthies. *Obstacle Detection*, pages 543–549. Springer US, Boston, MA, 2014. ISBN 978-0-387-31439-6. doi: 10.1007/978-0-387-31439-6_52. URL https://doi.org/10.1007/978-0-387-31439-6_52.

[64] Larry Matthies, Mark Maimone, Andrew Johnson, Yang Cheng, Reg Willson, Carlos Villalpando, Steve Goldberg, Andres Huertas, Andrew Stein, and Anelia Angelova. Computer vision on Mars. *International Journal of Computer Vision*, 75(1):67–92, 2007. ISSN 09205691. doi: 10.1007/s11263-007-0046-z.

[65] Kimberly Mcguire, Mario Coppola, Christophe De Wagter, and Guido De Croon. Towards Autonomous Navigation of Multiple Pocket-Drones in Real-World Environments. *Repository of Delft University of Technology*, 2017.

[66] Dongbo Min, Sunghwan Choi, Jiangbo Lu, Bumsub Ham, Kwanghoon Sohn, and Minh N Do. Fast global image smoothing based on weighted least squares. *Image Processing, IEEE Transactions on*, 23(12):5638–5653, 2014.

[67] J B Miog. Design of the locomotion subsystem for Lunar Zebro. Technical report, TU Delft, Delft, 2018.

[68] Bert et al. Monna. CP400.85 | Hyperion Technologies B.V., 2020. URL https://hyperiontechnologies.nl/products/cp400-85-processing-platform/?cn-reloaded=1.

[69] NASA. NASA Image and Video Library, 1997. URL https://images.nasa.gov/.

[70] NASA. Apollo 15 Images - Lunar Roving Vehicle, 2016. URL https://www.hq.nasa.gov/office/pao/History/alsj/a15/images15.html.

[71] NASA. Mars Pathfinder Rover, 2017. URL https://nssdc.gsfc.nasa.gov/nmc/spacecraftDisplay.do?id=MESURPR.

[72] NASA. Mars 3 Lander, 2018. URL https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=1971-049F.

[73] NASA. Lunokhod 1, 2018. URL https://solarsystem.nasa.gov/missions/lunokhod-01/in-depth/.

[74] NASA. Mars 2020 Perseverance Rover - NASA Mars, 2021. URL https://mars.nasa.gov/mars2020/.

[75] NASA and JPL. Mars Pathfinder Mission Web Site, 1997. URL https://mars.nasa.gov/MPF/.

[76] NASA's Mars Exploration Program. Driving Distances on Mars and the Moon, 2019. URL https://mars.nasa.gov/resources/6471/driving-distances-on-mars-and-the-moon/.

[77] Oon Ee Ng and Velappa Ganapathy. A novel modular framework for stereo vision. *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*, pages 857–862, 2009. doi: 10.1109/AIM.2009.5229904.

[78] *The OpenCV Reference Manual | StereoSGBM Class Reference*. OpenCV, April 2014.

[79] Edmund Optics. How Aberrations Affect Machine Vision Lenses, 2014. URL https://www.edmundoptics.com/resources/application-notes/imaging/how-aberrations-affect-machine-vision-lenses/.

[80] M. Osterman and G. B. Romer. History and evolution of photography. In *The Focal Encyclopedia of Photography, 4th Edition*, pages 23–57. Routledge, 2017.

[81] Mattijs Otten. *DeciZebro*. PhD thesis, Delft, University of Technology, 2018.

[82] Rafael Padilla, Sergio L. Netto, and Eduardo A.B. Da Silva. A Survey on Performance Metrics for Object-Detection Algorithms. In *International Conference on Systems, Signals, and Image Processing*, volume 2020-July, pages 237–242. IEEE Computer Society, jul 2020. ISBN 9781728175393. doi: 10.1109/IWSSIP48289.2020.9145130.

[83] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[84] Martin Peris. OpenCV: Stereo Matching Tuner, 2011. URL http://martinperis.blogspot.com/2011/08/opencv-stereo-matching.html.

[85] Sudeep Pillai, Srikumar Ramalingam, and John J. Leonard. High-performance and tunable stereo reconstruction. *Proceedings - IEEE International Conference on Robotics and Automation*, 2016-June:3188–3195, 2016. ISSN 10504729. doi: 10.1109/ICRA.2016.7487488.

[86] On Uma Pramote and Punpiti Piamsa-Nga. A stereo image matching method on images with varying light conditions. In *ICSEC 2015 - 19th International Computer Science and Engineering Conference: Hybrid Cloud Computing: A New Approach for Big Data Era*. Institute of Electrical and Electronics Engineers Inc., feb 2016. ISBN 9781467378253. doi: 10.1109/ICSEC.2015.7401439.

[87] P.D. Rajkakati. EuroMoonMars, 2020. URL https://euromoonmars.space/about/.

[88] Floris Rouwen. *Lunar Zebro, Software Design Of The Locomotion Sub-System With The Dezyne Model Driven Development Tool*. PhD thesis, Delft, University of Technology, 2018.

[89] S P Rovers. Literature Review. Technical report, TU Delft, Delft, 2019.

[90] U Saranli, M Buehler, and D E Koditschek. RHex: A Simple and Highly Mobile Robot. *International Journal of Robotics Research*, 20(7):616–631, 2001.

[91] Silvio Savarese. Computational Vision and Geometry Lab, Lecture 8 | Fitting and Matching, 2015.

[92] Davide Scaramuzza. 3D Robot Vision (ME47030) Lecture 6: Visual Odometry and SLAM, 2019.

[93] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Proceedings - IEEE Workshop on Stereo and Multi-Baseline Vision, SMBV 2001*, pages 131–140, 2001. doi: 10.1109/SMBV.2001.988771. URL www.middlebury.edu/stereo.

[94] Aravind Seeni, Bernd Schfer, and Gerd Hirzinger. Robot Mobility Systems for Planetary Surface Exploration – State-of-the-Art and Future Outlook: A Literature Survey. *Aerospace Technologies Advancements*, 2012. doi: 10.5772/6930.

[95] S. Serov and K. Shiryaev. StereoPi | Crowd Supply, 2017. URL https://www.crowdsupply.com/virt2real/stereopi.

[96] Kiran Venkateswara Sharma. *Adaptation Study of Zebro as Nano Rover for Lunar Exploration and Demonstration of Locomotion on Simulated Lunar Surface*. PhD thesis, Delft, University of Technology, 2018.

[97] Thomas Simko and Matthew Gray. Lunar Helium-3 Fuel for Nuclear Fusion : Technology , Economics , and Resources. *World Future Review*, pages 43–49, 2014. doi: 10.1177/1946756714536142.

[98] Spacebit. The New Economics of Space, 2020. URL `https://spacebit.com/`.

[99] P. D. Spudis, D. B.J. Bussey, S. M. Baloga, J. T.S. Cahill, L. S. Glaze, G. W. Patterson, R. K. Raney, T. W. Thompson, B. J. Thomson, and E. A. Ustinov. Evidence for water ice on the moon: Results for anomalous polar craters from the LRO Mini-RF imaging radar. *Journal of Geophysical Research E: Planets*, 118(10): 2016–2029, 2013. ISSN 01480227. doi: 10.1002/jgre.20156.

[100] Bill et al. Strickland. Mining The Moon - Rare Minerals - Helium 3, December 2004. URL `https://www.popularmechanics.com/space/moon-mars/a235/1283056/`.

[101] Sylvain Paris, Pierre Kornprobst, Jack Tumblin, and Frédo Durand. A Gentle Introduction to Bilateral Filtering and its Applications, 2008. URL `https://people.csail.mit.edu/sparis/bf_course/`.

[102] Richard Szeliski. Computer Vision : Algorithms and Applications. *Computer*, 5:832, 2010. ISSN 10636919. doi: 10.1007/978-1-84882-935-0. URL `http://research.microsoft.com/en-us/um/people/szeliski/book/drafts/szelski{_}20080330am{_}draft.pdf`.

[103] Dongjie Tan. Image Enhancement Based on Adaptive Median Filter and Wallis Filter. *Nceece*, pages 767–772, 2015. doi: 10.2991/nceece-15.2016.142.

[104] A. Thamizharasi and J.S. Jayasudha. an Illumination Invariant Face Recognition By Enhanced Contrast Limited Adaptive Histogram Equalization. *ICTACT Journal on Image and Video Processing*, 06(04):1258–1266, 2016. ISSN 09769099. doi: 10.21917/ijivp.2016.0183.

[105] Sjoerd Tijmons. Stereo Vision for Flapping Wing MAVs Design of an Obstacle Avoidance system. Technical report, Delft, University of Technology, 2012.

[106] Sjoerd Tijmons, Christophe De Wagter, Bart Remes, and Guido de Croon. Autonomous Door and Corridor Traversal with a 20-Gram Flapping Wing MAV by Onboard Stereo Vision. *Aerospace*, 5(3):69, 2018. ISSN 2226-4310. doi: 10.3390/aerospace5030069. URL `http://www.mdpi.com/2226-4310/5/3/69`.

[107] Qiaosong Wang, Zhan Yu, Christopher Rasmussen, and Jingyi Yu. Stereo vision–based depth of field rendering on a mobile device. *Journal of Electronic Imaging*, 23(2):023009, 2014. ISSN 1017-9909. doi: 10.1117/1.jei.23.2.023009.

[108] Y. Wang, M. Peng, K. Di, W. Wan, Z. Liu, Z. Yue, Y. Xing, X. Mao, and B. Teng. Vision based obstacle detection using rover stereo images. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, 42(2/W13):1471–1477, 2019. ISSN 16821750. doi: 10.5194/isprs-archives-XLII-2-W13-1471-2019.

[109] Garima Yadav, Saurabh Maheshwari, and Anjali Agarwal. Contrast limited adaptive histogram equalization based enhancement for real time video system. *Proceedings of the 2014 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2014*, pages 2392–2397, 2014. doi: 10.1109/ICACCI.2014.6968381.

[110] Dai Yiruo, Wang Wenjia, and Kawamata Yukihiro. Complex ground plane detection based on V-disparity map in off-road environment. *IEEE Intelligent Vehicles Symposium, Proceedings*, pages 1137–1142, 2013. doi: 10.1109/IVS.2013.6629619.

[111] Guoshen Yu and Jean-Michel Morel. ASIFT: An Algorithm for Fully Affine Invariant Comparison. *Image Processing On Line*, 1:1, 2011. ISSN 1314-2704. doi: 10.5201/ipol.2011.my-asift. URL `https://www.kinetica.mobi/que-es-el-mobile-marketing/`.

[112] Zhengyou Zhang. A Flexible New Technique for Camera Calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11), 2000. URL `papers://90b07e07-903c-45d4-a899-2629f88b6b69/Paper/p2444`.

[113] W. Zuo, C. L. Li, Z. B. Zhang, X. G. Zeng, Y. L. Zou, and G. L. Zhang. Scientific Data and its Release of Chang'e-3 Mission. *47th Lunar and Planetary Science Conference*, pages 2–3, 2016. doi: 10.1038/ngeo2474.