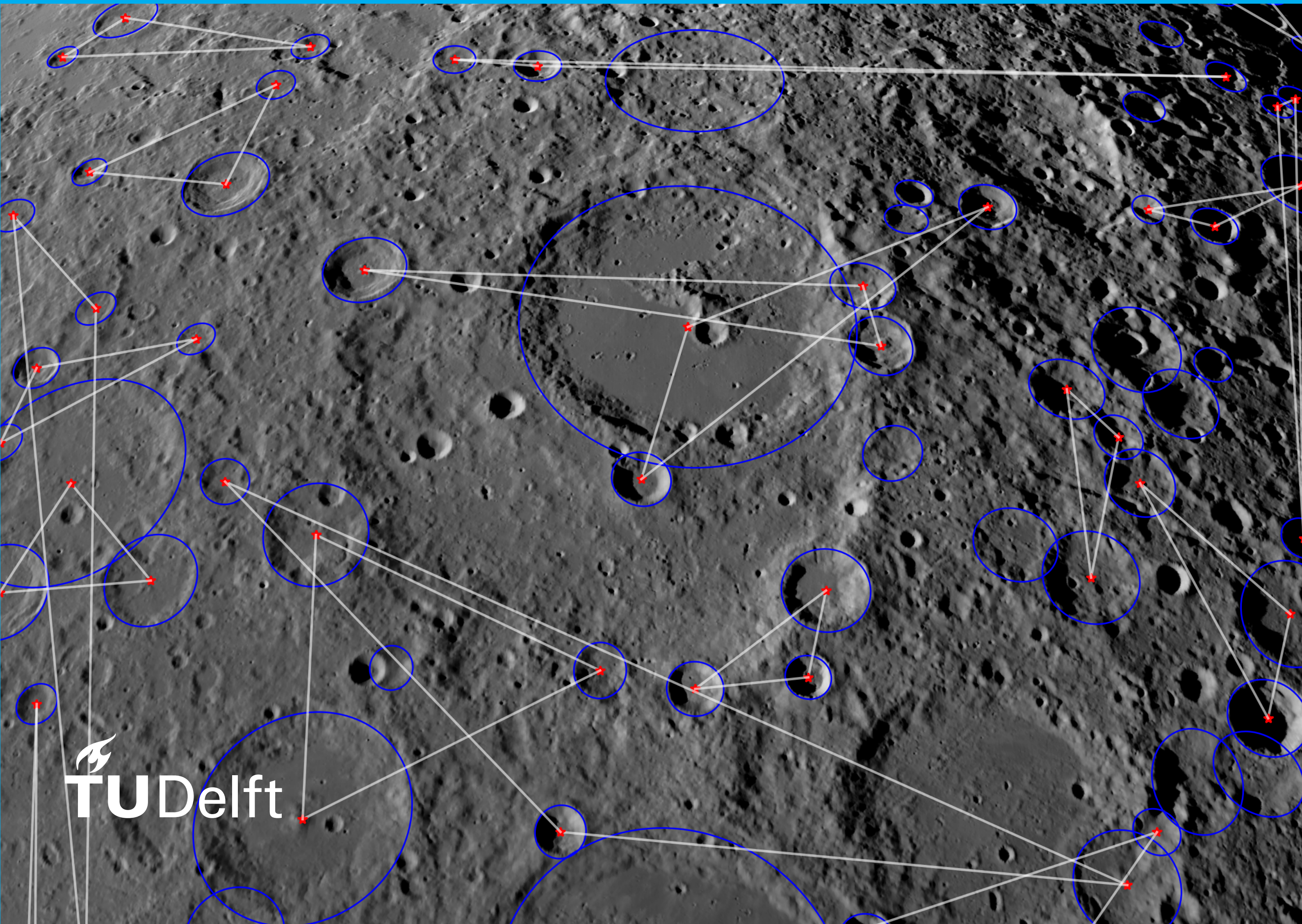
Autonomous Lunar Orbit Navigation With Ellipse R-CNN

Thesis Report

W. T. Doppenberg



TU Delft

Autonomous Lunar Orbit Navigation With Ellipse R-CNN

Thesis Report

by

W. T. Doppenberg

in partial fulfilment of the requirements for the degree of

Master of Science
in Aerospace, Aeronautical and Astronautical Engineering

at the Delft University of Technology,
to be defended publicly on Wednesday July 7, 2021 at 14:00.

Student number:	4362853	
Supervisor:	Dr. A. (Alessandra) Menicucci	TU Delft
Thesis Committee :	Dr. E. K. A. Gill (Chair)	TU Delft
	Dr. M. A. Kenworthy	TU Delft
	Dr. A. K. Dunne	Ubotica Technologies

Preface

I would like to thank my supervisor Alessandra Menicucci of TU Delft immensely for taking the time to help me with what for me has been an awesome project. Your advice has been invaluable. Putting me in touch with the right people at the right time, telling me when and when not to continue down a certain (research) path, and encouraging me to chase lofty ambitions for this project have now resulted in the work that lays before you. I could not be more grateful.

I'd also like to thank Jeremy Lebreton from Airbus DS, who has helped me greatly in setting up the simulation environment using SurRender. You really captured my thought process and freely offered advice. Furthermore, I would also like to thank Aubrey Dunne and David Moloney from Ubotica. You have offered insightful feedback and advice concerning my project. I would have hoped to have done more with the Neural Compute Stick, but I shall make it a hobby project of mine to deploy my custom model onto it - I will keep you in the loop.

I could not have achieved all of this without the help of my parents. It may have been a bit hard to verbalise what it exactly is that I've studied beyond "something with to do with space", but you have nonetheless supported and encouraged me at every step along the way. I hope that you know that I've appreciated your support for all these years.

For my roommates, friends, and Menghua, I believe I owe an apology for talking about the Moon too often. I promise I'll take a small break from it. Jokes aside, you always supported me along the way, and I'd like to let you know that I won't be sitting in my bedroom for days on end anymore.

W. T. Doppenberg
June 23, 2021

Contents

List of Figures

List of Tables

Glossary

Acronyms

AI	Artificial Intelligence	xiii, 1, 4 f., 9, 11 ff., 18 f., 28, 34, 36, 59, 61–64
ALHAT	Autonomous Landing and Hazard Avoidance Technology	2, 12
AP	Average Precision	37
AUC	Area Under the Curve	37
CDA	Crater Detection Algorithm	xiii, 4–7, 9–12, 15, 17, 19, 28, 31, 34 ff., 38, 41, 46, 49, 51, 55, 57, 63 ff.
CNN	Convolutional Neural Network	4 f., 7, 11, 31
COTS	Commercial-Off-The-Shelve	9
CV	Computer Vision	1, 4, 11, 18, 41
DEM	Digital Elevation Map	2, 4, 6, 22, 27, 51, 63, 65
DL	Deep Learning	4 f., 7, 9, 12, 16, 31, 34, 61, 63, 65
EKF	Extended Kalman Filter	xiii, 2, 6, 8, 18, 20, 46, 55 ff., 64, 71
ESA	European Space Agency	2, 11
GNC	Guidance Navigation & Control	13, 17, 59, 64
HIL	Hardware-in-the-Loop	59, 61, 65
HITL	human-in-the-loop	1
IMU	Inertial Measurement Unit	1, 65
IoU	Intersection-over-Union	32, 36 f.
JAXA	Japan Aerospace Exploration Agency	22
JPL	Jet Propulsion Laboratory	2
KPI	Key Performance Indicator	51
LIS	Lost-In-Space	xiii, 3, 8 f., 11 ff., 18, 41, 43, 45, 51, 61, 64
LOLA	Lunar Orbiter Laser Altimeter	22
LoS	Line-of-Sight	21
LRO	Lunar Reconnaissance Orbiter	22
LSTM	Long Short Term Memory	65
MCMF	Moon-Centred-Moon-Fixed	21 f., 24
ML	Machine Learning	4 f.
MLP	Multi-Layered Perceptron	5, 7, 31, 33
NASA	National Aerospace & Space Administration	2, 22 f.

NCS	Neural Compute Stick	61
NN	Neural Network	5, 61
PDS	Planetary Data System	22
RANSAC	Random Sample Consensus	xiii, 8, 41, 45, 56
RoI	Region of Interest	31, 33
RPN	Region Proposal Network	7
SEE	Single-Event Effect	14, 59
TID	Total Ionising Dose	14 f., 59
TRL	Technology Readiness Level	13, 16, 18 ff., 34, 61 f.
TRN	Terrain-Relative Navigation	xiii, 1 ff., 6–9, 11–14, 16, 19 f., 28, 49, 51, 55, 61–65
VPU	Vision Processing Unit	3, 12

Abstract

The resurgence of interest in landing on the Moon has sparked the creation of a number of novel technologies concerning Terrain-Relative Navigation (TRN) systems [12, 25]. They aid in the need for increasingly precise landing, as well as enabling autonomous goal-oriented autonomy [26]. To achieve this, most technologies use a ubiquitous feature present on the Moon: impact craters. Using databases containing of over a million labeled impact craters [71], research has yielded end-to-end algorithms that allow for the detection of craters in a camera image and subsequent self-localisation. These types of algorithms generally use a technique called crater pattern matching, where an indexable feature is generated from the detected set of craters to find correspondence between the spacecraft state and the camera input. Recent developments concerning crater pattern matching techniques have yielded novel methods using relations from the theory of projective geometry [17]. These techniques require crater rims in the image plane to be fit with ellipses as accurately as possible.

This research describes the design, development, and testing of an end-to-end TRN system demonstrator. This system demonstrator utilises a novel region-based Crater Detection Algorithm (CDA) together with a projective invariant-based crater pattern matching technique that allows for robust ego-position estimation The system's goal was to mark an improvement over earlier attempts to build an end-to-end TRN in terms of robustness, performance, and accuracy. This has been achieved by the development of a tightly integrated system developed specifically for the task of ego-position estimation from a Lost-In-Space (LIS) situation.

Using physically accurate data generated using specialised space exploration rendering software [67], a region-based object detection model based on Ellipse R-CNN [23] was built to directly fit ellipses to impact craters in the image plane. These parameterised ellipses were subsequently used to generate a seven-element descriptor per crater triad for direct matching with a scalable database. Using the selenographic position and shape of matched craters, a system of equations is solved using Random Sample Consensus (RANSAC) [30] to achieve a robust state estimation system. Tests show that the proposed system's measurements alone combined with an Extended Kalman Filter (EKF) [88] result in an error of less than 160 [m] from a heavily degraded (> 500 [km]) initial state.

The development of this demonstrator yielded three advancements to the field of TRN: a flexible data generation pipeline, a novel Artificial Intelligence (AI)-based CDA, and the first results of a region-based detection model in tandem with a modern crater pattern matching technique. The code is publicly available[1], allowing future projects to improve upon the current state of the demonstrator.

[1]`https://github.com/wdoppenberg/crater-detection`

1 Introduction

During the later Apollo missions, the astronauts on-board the Lunar Landing module had to ensure a soft landing at a safe location which was level and devoid of hazards. To achieve this, the descent stage of the Lunar Module used a human-in-the-loop (HITL) approach, combining automated systems and human interaction to perform the required actuations to reach the predesignated landing spot relatively accurately [48]. Figure 1.1 displays a schematic overview of the components that make up the control loop, clearly showing that the interface between the astronaut and the Flight Computer system is present in both feedback (Inertial Measurement Unit (IMU), data display, optics) and control (manual control signals). At the time, a HITL-based control loop was an obvious choice considering the resources that were available.

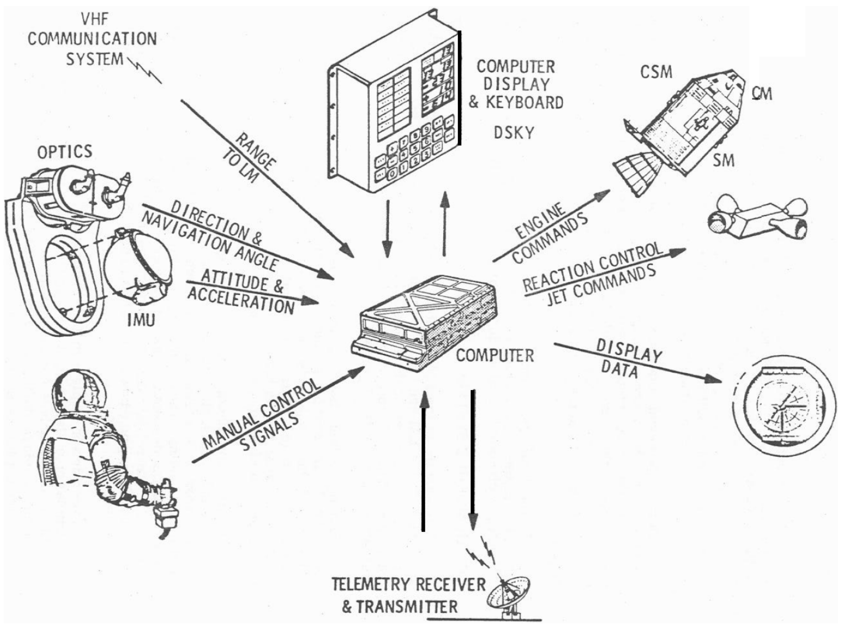


Figure 1.1: Apollo 11's Lunar Module Guidance Navigation & Control (GNC) system schematic [48].

The majority of control systems for spacecrafts have been based on solutions using linear algebra and linearisation [29], and have served their purposes well in the past. However, increasingly complex mission demands require avionics engineers to look into non-linear designs, as this type of technology has the potential to surpass current technological limitations. The precision landing problem can benefit greatly from this way of thinking, considering the promise of Computer Vision (CV) or Artificial Intelligence (AI) shows in general to such problems. The non-linear nature of the relation between a view of the Lunar surface and the relative position can be potentially be taught by setting a target objective and letting the system train itself. The main downside of these types of models is the significant processing power and time required. However significant advances in processing hardware have made it possible to introduce non-linear control systems for spacecraft. Combined with a renewed interest in planetary exploration using robotic landing vehicles, a flexible, cost-effective, and self-contained autonomous position determination system can expand the possibilities of future missions greatly. Current solutions use a combination of inertial, visual, and radiometric measurement updates; the focus of this document is on the visual navigation subsystem, or Terrain-Relative Navigation (TRN).

1.1. Terrain-Relative Navigation

Navigating towards a precise landing spot requires state knowledge throughout the mission envelope. For example, the Autonomous Landing and Hazard Avoidance Technology (ALHAT) project was initiated by National Aerospace & Space Administration (NASA) Jet Propulsion Laboratory (JPL) to design a TRN system that could guide a lander within 100 [m] of its predetermined target on the surface of the Moon [50]. One recent example of such a system was on-board of the Chang'E-4 unmanned Lunar lander [56]. This system used photogrammetry to find a visual correspondence between its pose and the predetermined landing spot.

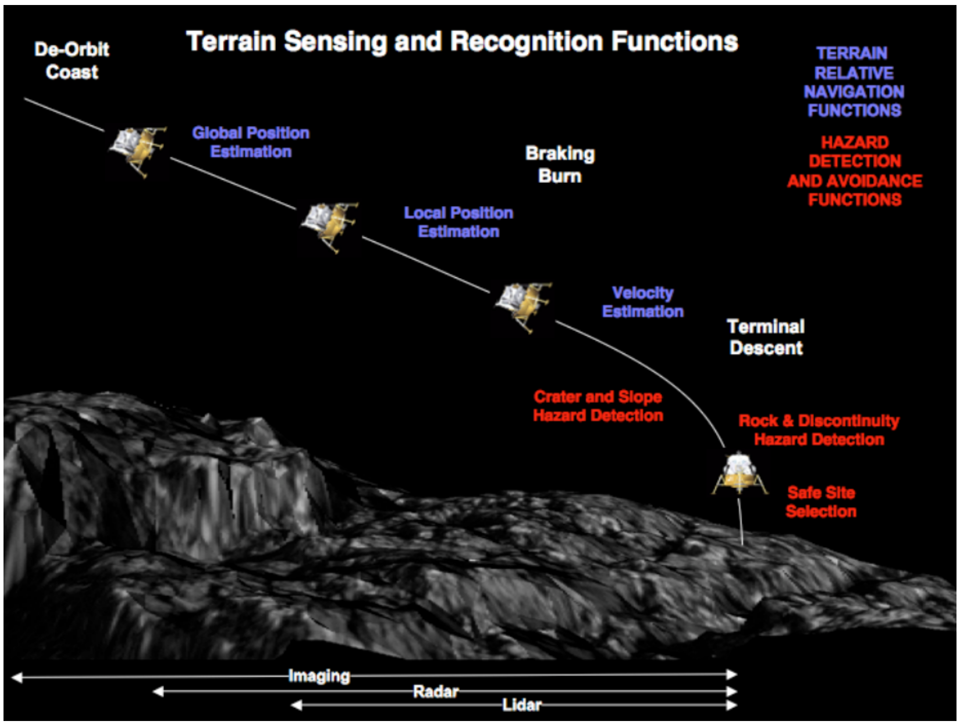

Terrain Sensing and Recognition Functions
De-Orbit Coast
Global Position Estimation
Braking Burn
Local Position Estimation
Velocity Estimation
Terminal Descent
TERRAIN RELATIVE NAVIGATION FUNCTIONS
HAZARD DETECTION AND AVOIDANCE FUNCTIONS
Crater and Slope Hazard Detection
Rock & Discontinuity Hazard Detection
Safe Site Selection
Imaging
Radar
Lidar

Figure 1.2: Illustration of TRN in a precise lunar landing context, reprinted from [50].

State estimation can be achieved through global position estimation, local position estimation, and velocity estimation. The first two require prior knowledge about the landing site, for example a Digital Elevation Map (DEM) of the (wider) area to find correspondence between sensor input and an on-board database. The latter however, is a less precise method of local position estimation because of the error accumulation in velocity measurements. Combining these types of measurements is ideal, as it can be fed into a state estimation algorithm like an Extended Kalman Filter (EKF), or a modern equivalent.

1.1.1. Autonomous Navigation

TRN is one manifestation of autonomy in space systems. As illustrated by the many functionalities described in Figure 1.2, autonomy is achieved using a system which - given the current location and situation - intelligently utilises multiple instruments to achieve a set goal. In the case of ALHAT the goal is to achieve highly precise autonomous landing. By the definition of the European Space Agency (ESA), this would constitute a mission execution autonomy level E3 [26]. The spectrum of these levels are given in Table 1.1 which provides a clear definition of the relation between the degree of ground intervention and autonomy.

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

Table 1.1: Mission execution autonomy levels, reprinted from [26].

Essentially, the highest degree of autonomy implies the usage of a highly adaptable navigation system. This system should be capable of utilising on-board sensors input to autonomously achieve its goal without ground intervention.

1.1.2. State of the Art

Most recent implementations of TRN yield two types of solutions: Active & Passive. Active methods use an active emitter directed at the Lunar surface, capturing the reflected signals to determine the surface geometry. This has the benefit of working in any lighting condition but is generally more complex, expensive, and usually only works at low altitudes. Passive methods use a camera system to capture the (sunlit) surface of the Moon, identifying either absolute or relative position estimations. The benefit of this method is that it is relatively inexpensive, works at any altitude, and utilises components that can serve multiple purposes, such as a camera or Vision Processing Unit (VPU). However, the main downside is that this requires a sunlit terrain, constraining mission planning flexibility. Table 1.2 gives a condensed overview of different types of TRN mostly based on the survey provided in [50], with the addition of extra sources of recent developments in this domain.

Type	Sensor	Approach	Input	Strengths	Weaknesses	Sources
Passive	Camera	Crater pattern matching	Crater database; camera output; attitude	Robust against changes in illumination; space & time efficient; does not require nadir-pointing; works at any altitude	Requires sunlit terrain; requires terrain with craters; requires attitude input	[17, 21, 24, 40, 79, 87]
		SIFT pattern matching	SIFT database; camera output	No attitude or altitude information required; applicable with all types of terrain	Requires sunlit terrain; changes in illumination degrade performance; sensitive to out-of-plane rotations	[57, 84]
		Image to global map correlation	Surface map; camera output; attitude; altitude	Applicable with all types of terrain; requires a single image without post-processing	Requires sunlit terrain; changes in illumination and terrain relief degrade performance; time & space inefficient	[14, 77]
Active	LiDAR	Shape signature pattern matching	LiDAR output; motion data; shape signature	No prior state knowledge required; No sunlight required	Expensive sensor; requires distinguishable terrain relief; less mature than camera; time inefficient	[32, 49]
		DEM correlation	LiDAR output; motion correction data; attitude; DEM	No sunlight required; More robust than altimeter	Requires complex scanning array; less mature than camera	[74]
	Altimeter	DEM correlation	Altimeter output; motion correction data; attitude; DEM	No sunlight required; works at higher altitudes than LiDAR	Requires long contour; less mature than camera	[36]

Table 1.2: A comparison between TRN approaches

Most research into TRN algorithms has been focused on passive techniques, especially crater pattern matching, as is illustrated by the amount of sources available describing solutions using this method. The reason for this popularity is clear: any lander design foresees the installation of a camera module which can be repurposed to serve multiple uses, including TRN. However, due to the complex relation between surface type, geometry, and the sun angle, it has proven difficult to create a generalisable solution to detect craters. Furthermore, designing an end-to-end Lost-In-Space (LIS) crater pattern matching TRN system has not been achieved. It is herein that the opportunity lies to expand the body of knowledge concerning navigation algorithms, and to realise fully autonomous navigation above a large crater-filled body such as the Moon.

1.2. Crater Detection Algorithms

The study of impact craters has historically been the subject of planetary science subjects, for example on the topic of whether or not the Late Heavy Bombardment occurred [42]. This brought about the creation of crater databases which stored the location and shape of impact craters [46, 71, 75], which in some cases contain more than a million entries. Craters are therefore suitable to navigate low orbit since they can be detected and quantified as basic geometric shapes such as circles or ellipses. This is the reason the past decade has yielded a number of proof-of-concepts for TRN using craters [24, 58, 63], most of which could be described as a system consisting of two parts: a detector and an identifier.

Crater Detection Algorithms (CDAs) have been developed to work on either DEMs for aforementioned scientific purposes, or on optical imagery for state estimation. The former requires that the surface of the primary body (e.g. the Moon) is scanned using a laser altimeter or similar instrument, whereas for state estimation a camera is often considered to be the sensor of choice as this holds significant benefits in cost and complexity. The drawback of optical imagery is that the input data heavily depends on exogenous variables such as sun angle, spacecraft orientation, surface types, and more. Current CDAs can be split into two groups: traditional and AI-based solutions.

1.2.1. Traditional Computer Vision

Traditional algorithms are often called as such because the concepts required to build them stem from an era before AI-based solutions, with methods like edge detection, filtering, and thresholding. Their logic consists of a string of manually engineered operations that derive crater shapes in the most robust way possible. Figure 1.3 shows an example of a traditional CDA by [60], where crater rims are detected using techniques such as Canny Edge detection [13, 27, 89] combined with highly coupled pre- and post-processing operations.

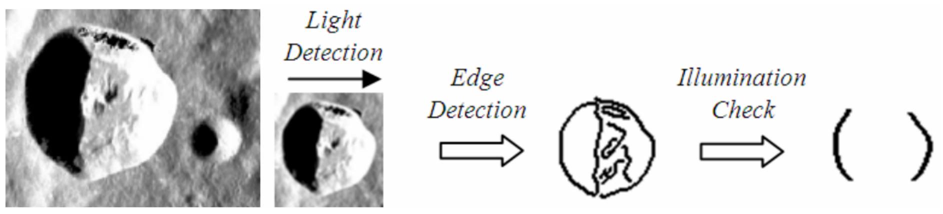


Figure 1.3: Example of a traditional CV crater detection algorithm, reprinted from [60].

Inherently, this high coupling of functions to derive crater shapes does not generalise well for varying viewing angles, noise, or lighting conditions. For example, if lighting conditions were to change significantly, the appearance of a crater in the pixel frame would change dramatically too. Advanced solutions using these techniques that try to mitigate the shortcomings of traditional crater detection have been built [41, 43], but nonetheless the focus of most recent CDA research has shifted to AI-based solutions.

1.2.2. Deep Learning-based CDA

CDAs using DL techniques have the potential to yield more robust solutions, as they self-engineer the non-linear relations required to detect crater shapes through a training process (see section 1.3). This novel technique requires an introduction before its application to CDAs can be fully understood.

1.3. Deep Learning

Recent years have seen novel solutions based on DL appear for a host of domains because of a surge in dataset size and available computing power [62]. Especially the CV domain has benefited from the usage of Convolutional Neural Network (CNN)-based models, since the unstructured and highly dimensional nature of image data lends itself well to these types of architectures. The significant processing requirements can be offset by deploying these models on massively parallel processors, which have only recently been deployed in a spacecraft platform [6, 33].

1.3.1. Concepts

It is a common agreement that the following holds for the relationship between Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL):

$$\text{AI} \supset \text{ML} \supset \text{DL} \tag{1.1}$$

There are many exceptions, however it helps to illustrate the relationship between these terms as they are often used interchangeably in papers that use this method for CDAs. The goal of any Neural Network (NN) (also referred to as Multi-Layered Perceptron (MLP) or deep feed forward network) is to approximate some function f^* by learning a set of parameters θ that result in the best function approximation $f(\mathbf{x};\theta)$ [37]. In short, the objective is defined as:

$$f(\mathbf{x};\theta) \approx f^*(\mathbf{x}) \tag{1.2}$$

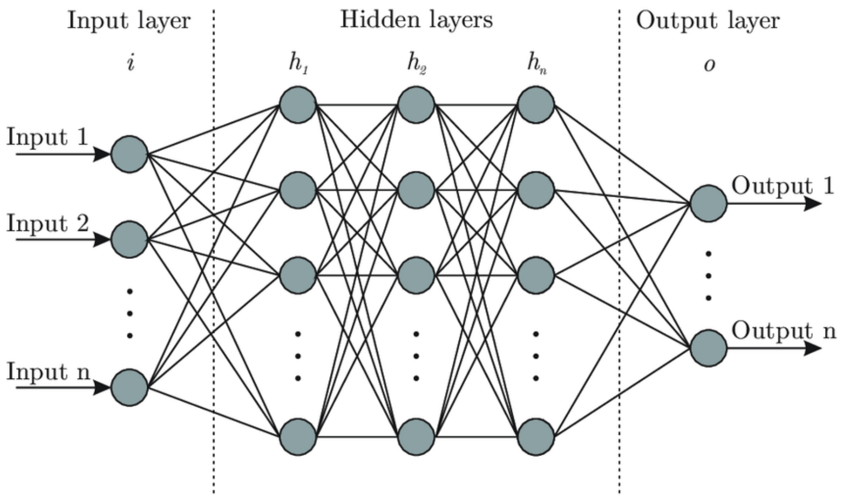


Figure 1.4: A general representation of a Neural Network, reprinted from [1]

These NNs are called *networks* because they comprise multiple (non-)linear functions that form layers, as shown in Figure 1.4. For example, if $f^{(1)}$, $f^{(2)}$, and $f^{(3)}$ are chained together to form a NN, it becomes:

$$f(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x}))) \tag{1.3}$$

Specialisations of NN also exist in the form of CNN, which limits the amount of connections that layers have, meaning that deeper networks are possible without the curse of dimensionality incurring heavy performance penalties. This is by replacing the general matrix multiplication that occurs between layers in a fully connected NN with convolution. Convolution is the process where the network is only sparsely connected, resulting in less connections while maintaining the ability to learn meaningful features with kernels that occupy only a few tens or hundreds of pixels. For image processing, this is generally the preferred strategy because of the input size, and because "*units in the deeper layers may indirectly interact with a larger portion of the input ... this allows the network to efficiently describe complicated interactions between many variables by constructing such interactions from simple building blocks that each describe only sparse interactions*" [37].

Training any NN consists of using a data set $\{\mathbf{x}, \mathbf{y}\}$ which can - by updating θ using a method such as Gradient Descent - yield a model that exhibits behaviour that approximates Equation 1.2. In the case of a CDA, this usually means per-pixel classification of crater rims, with some post-processing steps added to get the crater shape.

1.3.2. Image Segmentation

DL models have mostly been applied to image segmentation tasks. These tasks generally aim to classify an image's content, be it image classification or segmentation. The former attempts to label images as a whole, whereas the latter is tasked with classifying each pixel on whether it belongs to a certain class (and instance). Commonly used generalised open-sourced datasets include ImageNet [22] (image classification),

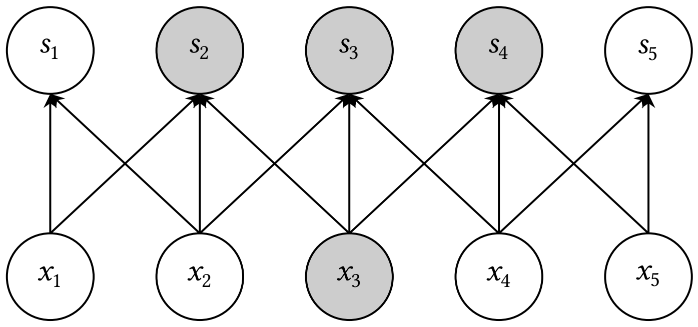


Figure 1.5: An illustration of the concept behind convolution, inspired by [37].

and CoCo [55] (semantic segmentation). Image segmentation models are therefore excellent candidates for a modern CDA, potentially being capable of a higher degree of generalisation. Up until recently, semantic segmentation has generally been the preferred method of choice for CDA because of a popular implementation called DeepMoon [78]. The main concern of this type of model is that additional post-processing steps are required to distinguish individual craters in the detection mask. More recent research into applying instance segmentation models for detecting craters has yielded promising results [28].

Semantic Segmentation
Semantic segmentation is the process of labeling each pixel in an image. One application is biomedical image segmentation, with U-Net being a popular example [73]. As a matter of fact, this exact architecture has been applied to crater detection, and was renamed DeepMoon [78]. DeepMoon was first conceived as a novel CDA for the purpose of generating a crater catalogue using DEM data of the Moon. This model uses a convolution - deconvolution setup that copies feature maps at corresponding feature sizes. This allows the network to infer high resolution segmentation by propagating context information from preceding layers. The layout of the architecture is shown in Figure 1.6. DeepMoon used the architecture to generate per-pixel segmentation of crater rims, after which a template matching step iteratively fits circles on the mask to find crater instances. This method is not necessarily fast, but in the context of DeepMoon's use case (crater cataloguing) it was considered adequate.

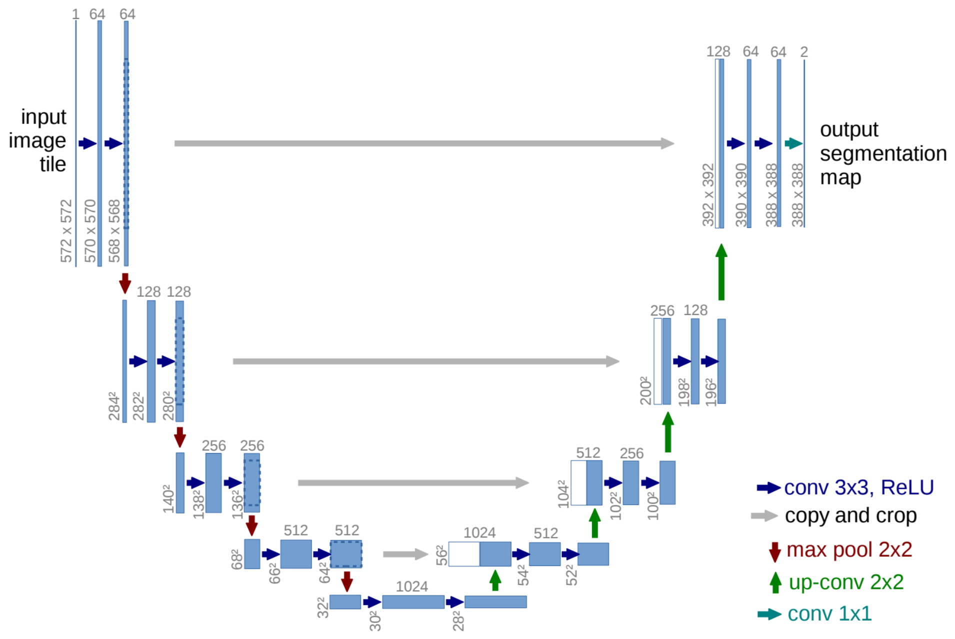


Figure 1.6: U-Net architecture, reprinted from [73].

For TRN, an iteration of DeepMoon trained using images with simulated camera images combined with an

EKF named LunaNet was developed [24]. Instead of using template matching, this model uses the prediction mask in combination with several post-processing steps to form a set of fitted ellipses per image. The consequent crater identification is done with the assumption that prior approximate state information is available during the crater identification step.

Instance Segmentation

Compared to a semantic segmentation, instance segmentation identifies distinct objects belonging to a class in an image. This has the intrinsic benefit of having positional information as well as shape (usually returned as a bounding box) for each detected object in an image. It is obvious that this type of model has great potential for crater detection, having the benefit of replacing post-processing steps required when using per-pixel classification masks such as the output of DeepMoon.

Instance segmentation is essentially an instance detection model with extra predictors added. The *backbone* that supports instance detection models are based on deep CNNs image classification models such as VGG [80] or ResNet [44]. These models can be up to 152 layers deep, and have typically been developed to work well on a general image classification dataset called ImageNet [22]. For this task, the network uses a method called end-to-end learning, which refers to a model learning the feature engineering required to retrieve meaningful intermediate representations of the data. This essentially means that traditional image manipulation techniques do not need to be implemented for the model to function. These convolutional layers act as filters on an input image, and train themselves to highlight spatial relationships that aid in the classification (or detection) process.

The *target* data for an object detection model consists of instances of the objects to be detected, and are captured (mainly) by two variables: its *label* and the *bounding box*. The label is dependent on the amount of classes that are included in the dataset, and the bounding box is the pixel location of the object in the image delimited by a rectangle described as $[x_{\min}, y_{\min}, x_{\max}, y_{\max}]$ (see Figure 1.7). In the case of a CDA this means that there are two classes: background and crater. The model's capability to distinguish objects from the background is quantified by its *objectness* score.

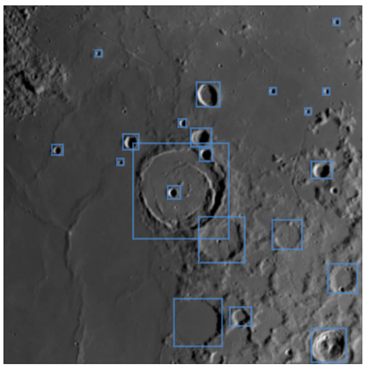


Figure 1.7: Illustration of bounding boxes fitted to craters in the image plane.

R-CNN [35], or Regions with CNN, was a novel approach to object detection and classification. It uses extracted region proposals from an input image as input to a CNN for classification. This means that the model architecture can return multiple instances with associated class predictions. Two newer iterations followed this, aiming to improve training and testing speed: Fast R-CNN [34], and Faster R-CNN [69]. The latest iteration introduced Region Proposal Networks (RPNs), which leverages the learned weights present in a state-of-the-art image classification network to generate region proposals. This only marginally increases the cost of computing proposals for a single image.

Region-based models have very recently been applied to CDA, showing promising results [28]. Utilising a classification back-end to discern individual craters from the image is a powerful method, as it can be trained

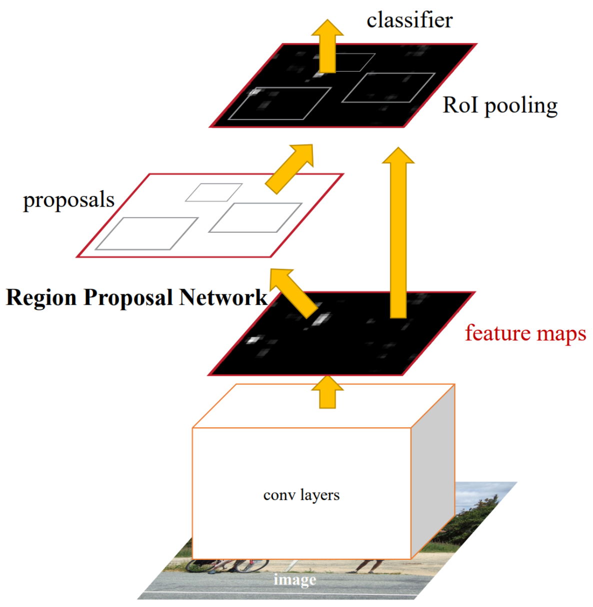


Figure 1.8: Overview of Faster R-CNN, reprinted from [69].

to work in varying circumstances caused by lighting conditions, spacecraft attitude, etc. However, this model was designed for crater cataloguing, not TRN. Only having bounding box predictions of craters does not yield enough information to fit ellipses to crater rims. However, an existing model called Mask R-CNN [45] can for example perform per-pixel classification per instance called instance segmentation. Using pixel-wise masks for target instances, an extra predictor can be trained to perform pixel-wise classification based on the latent features present in the backbone. This predictor is generally a relatively simple MLP with 3 layers.

Opportunities

It is clear that the aforementioned architectures have great potential for crater detection. The fact that these types of models can be trained to perform instance segmentation tasks triggers the question whether the same type of regressors can be used to predict (normalised) ellipse parameters per detected instance. Further research and development must answer this question, and is thus included in the scope of this project.

1.4. Crater Pattern Matching

Using detected landmarks such as craters, a spacecraft can determine its position above its primary body (such as the Moon). First used by the Near Earth Asteroid Rendezvous (NEAR) mission [15] to navigate around Eros. Craters prove to be an easily quantifiable feature because of the basic geometric shape (an ellipse) used to describe its rim. There is one main assumption to make when designing an autonomous orbit determination system: whether or not prior approximate state knowledge is present or not. Not having such information available at the time of execution significantly impacts the complexity of the problem, illustrated by the name: LIS. This term is similarly used when designing star tracker algorithms, which have the capability to reinitialise the navigation state estimate without prior knowledge [70].

Crater pattern matching offers a space- and time-efficient alternative to more costly algorithms such as image correlation techniques. Building an image correlation database for a LIS ego-position estimation algorithm would set a significant processing requirement to be able to work in real-time. Considering the limited (processing) power available on a spacecraft, this is undesirable. Instead, geometric relationships can be made between the apparent elliptical shapes of the rims of impact craters, generating discriminating features represented by a numerical descriptor that can be used to find correspondence between the camera input and an on-board database. The high amount of classified impact craters available in public datasets such as [71] allow the creation of a global crater identification database, with the goal of expanding TRN functionality.

1.4.1. With Available State Information

[11, 18, 79] describe another complete TRN system using crater pattern matching. This system requires a-priori state information combined with crater shape matches to determine a translation vector $\delta\boldsymbol{x}$ that solves the crater-camera homography. This homography is the affine transformation between the craters in the image plane and selenographic space. To solve this, it utilises a Random Sample Consensus (RANSAC)-type algorithm, which is a method for solving systems of linear equations with a high amount of outliers present. The system has been tested using synthetic images as well as with a physical mock-up of the Lunar surface, and proved to work well. Similarly, LunaNet's crater identification subsystem uses an adapted version of this technique, using the output of an EKF to find crater correspondence.

1.4.2. Lost-In-Space Algorithms

[40] is an example of a Lost-In-Low-Lunar-Orbit algorithm, and utilises an approach derived from star tracker algorithms. In summary, it uses the triangles formed by triads of craters projected into the image plane to form a unique descriptor based on the two smallest inner angles. A fourth crater is then located in the image, which is then used for a second database search with the newly formed crater triangle. The probability that said match is false is used to then either reject or accept the identification. The problem with this method is that it is not rotation invariant, meaning that only nadir-pointing camera angles are allowed. Furthermore, this method uses just 2 consecutive searches for 2 values to discriminate between craters, which works acceptably well for the amount of craters used in this research (4117), but does not scale well with a larger crater dataset.

[63] present a method for crater matching using a concept called projective invariants. This is a property of pairs of geometric shapes, like ellipses, to be characterised by a function that yields the same outcome regardless of any affine transformation that are applied. An example of such a transformation is the aforementioned crater-camera homography. Using this property, a seven-element descriptor can be made per coplanar crater triad to use for a global crater index. Since this method uses projective invariants, it also means that it can work under off-nadir pointing attitudes, increasing flexibility. [17] expands this method to non-coplanar triads on non-degenerate surfaces (e.g. a spheroid), and introduces an algorithm for using matched crater shape information and a known attitude input (from e.g. a star tracker) to estimate the spacecraft's position through solving a linear system of equations. This is a truly LIS algorithm, but is yet to be paired with a modern iteration of a CDA. As mentioned in [17]: "*Our crater identification algorithm is critically dependent on the localization of the best-fit ellipse to the crater rim, which is often not the metric used to design or evaluate CDAs. Thus, finding (or developing) an appropriate CDA is an obvious topic of follow-on work*". This presents an opportunity to create a tightly integrated LIS TRN algorithm by developing both the CDA and a projective invariant-based crater identification algorithm in parallel.

1.5. Research

The need for precise autonomous landing combined with advances in Commercial-Off-The-Shelve (COTS) hardware accelerators for mobile applications creates opportunities to explore novel vision-based solutions for low orbit navigation problems. To ensure that the most knowledge is gained, the domain for the CDA is restricted to AI-based solutions on purpose. This enables the research to be focused, to assess a real-world application of specialised vision processing chips in spacecraft, and to attempt to solve an open problem in the spacecraft avionics domain.

Developing, training, and verifying an AI-based TRN system requires the creation of a robust testing environment that is capable of generating large amounts of data. Current scene simulation technologies provide novel ways of approaching this, and could prove to be useful should further research be performed on similar problems. Furthermore, marrying the rapid technological progress of the DL domain with the relatively conservative space engineering domain may prove to be difficult, but may hold significant benefits for exploring alternative methods for solving vision-based navigation problems. And finally, using the most recent methods for achieving robust ego-position estimation with crater pattern matching in tandem with a CDA presents a unique opportunity to assess the current state-of-the-art in TRN techniques.

1.5.1. Research Questions

To guide the research, a set of (sub-)questions have been defined. These address the aforementioned opportunities:

RQ Is region-based object detection a suitable method for a Lost-In-Space Terrain-Relative Navigation system?

- **RQ1** What requirements should a dataset meant for developing, training, and verifying a Terrain-Relative Navigation system fulfil?
- **RQ2** Is a Crater Detection Algorithm using region-based object detection sufficiently robust for a Terrain-Relative Navigation system?
- **RQ3** How does an absolute position estimation algorithm perform in tandem with a Crater Detection Algorithm using region-based object detection?

1.5.2. Research Objectives

To be capable of answering the research questions, a set of goals are given per research question. The goals essentially define the needed components and associated tests that need to be developed and implemented.

Answering **RQ1** requires the creation of a representative environment in which to verify the complete system, manifested as a dataset with labeled data along with metadata (e.g. camera position, attitude, etc.). This requires a method of simulating camera input along with the projected craters in the image plane (labels), as well as a method of generating a large enough dataset for training a region-based object detection model.

Answering **RQ2** requires developing a novel CDA using region-based object detection techniques. An assessment of the state-of-the-art of these types of models was done to find opportunities and to be able to formulate a analytical description of the required steps for implementation.

Finally, answering **RQ3** requires the implementation of a crater pattern matching and position estimation algorithm that can work with the output of the developed CDA. Again, this requires a study of available techniques, and subsequently implementing them. This subsystem will most likely consist of a kind of database and a ego-position estimation algorithm, hence these need to be implemented as well. Finally, the full system is verified in a representative environment. A summarised overview is given in Table 1.3.

Research Question	Goal	Sub-goal
RQ1	Create a dataset fit for developing, training, and verifying a TRN system.	Camera input simulation Crater projection from dataset Robust data generation pipeline
RQ2	Develop and implement a CDA using region-based object detection techniques.	Assess state-of-the-art Feature engineering Model training Evaluate subsystem
RQ3	Implement a crater pattern matching algorithm with position estimation functionality and test it in combination with the developed CDA.	Assess state-of-the-art Create indexable database Develop position estimation algorithm Evaluate system

Table 1.3: Research goals for the development of a TRN system using region-based object detection techniques.

2 Systems Engineering

It is proposed to design, develop, and test a demonstrator to answer the questions posed in section 1.5. Approaching the full demonstrator as a system in itself allows the research to be focused on achieving the highest amount of added knowledge on the chosen subject of TRN using region-based object detection. To this end, a list of requirements is generated along with methods for verification.

2.1. Front End Process

Following the steps from [52], the identified opportunity can be translated into a detailed system design. First, the active and passive stakeholders along with their capabilities and characteristics are defined. Following from their priorities and the set scope a system concept is chosen, illustrated by the system context diagram. Next are the operational, functional, and physical views, from which the system's main objectives, functions, and layout can be derived.

2.1.1. Stakeholders

Answering the posed research questions has the potential to benefit space engineers involved in the design and manufacturing of the processing hardware that is used to power TRN systems. In fact, the proposed combination of a very modern iteration of CNNs (region-based object detection) with a LIS navigation to form an integrated avionics solution has never been performed before. Combining results from previous research into a single end-to-end open-sourced TRN system with modern techniques is a key step for avionics as it can serve as a starting point or benchmark for future research.

2.1.2. Need

The need for a system as proposed in this research is reflected in a Statement of Work issued by the European Space Agency (ESA) for AI techniques for spacecraft avionics:

> "One space application that could greatly benefit from application of non-linear systems is the autonomous vision-based navigation of spacecraft. Indeed, the demand of highly accurate relative navigation in a non-linear and unpredictable harsh environment including time constraints compels the designers to consider alternative methodologies and concepts. One of these ideas is to use adaptive systems with non-linear elements. Artificial Intelligence is a product of a non-linear adaptive system. The way it can be manifested depends on the technology level and the current knowledge." [29]

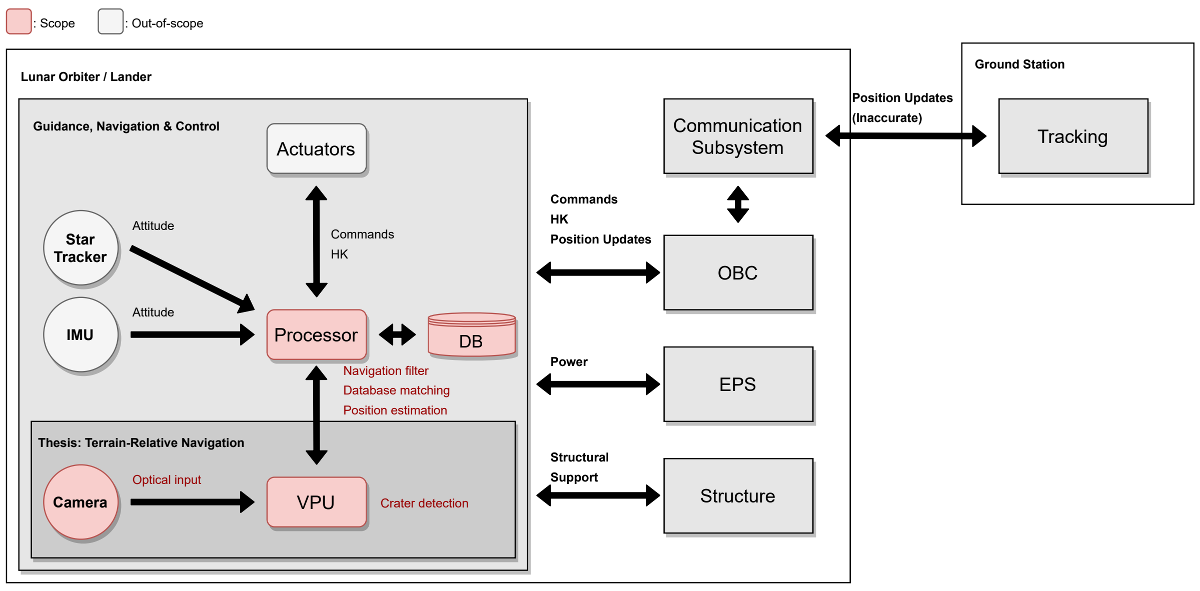
In summary, there is a clear demand for a highly accurate vision-based navigation solution incorporating AI. The development of such a solution serves to explore and assess the potential functional and performance benefits of AI in a spacecraft avionics system. Developing a navigation solution that functions in a LIS situation can be considered the ultimate form of adaptivity - another keyword in the above statement.

2.1.3. Opportunity & Implementation

The applicability of AI-based techniques, and region-based object detection specifically, along with increased processing power in a space avionics system present an opportunity to design and develop a technologically superior solution through technology fusion. These techniques have been transformative for many CV problems, and exploring the potential benefits for space applications is therefore a key component of this research. To test this claim, a demonstrator shall be developed to assess the performance of a region-based object detection CDA in combination with an ego-position estimation algorithm. Furthermore, the end-goal of this research is to make a demonstrator that has proven its capabilities in a software environment, and is ready to be ported to an embedded target for hardware performance testing. This requires the demonstrator to be platform-agnostic, and the results to be reproducible for other researchers.

2.1.4. System Context

Figure 2.1 displays a system context diagram that defines the setup for our TRN system, and is inspired by ALHAT [12]. Defining what components are available for a physical implementation of the system naturally affects the software implementation, and, as a result, the proposed demonstrator. The introduction of a reprogrammable (and thereby flexible) VPU into the system greatly influences the available design options for CDA design. Modern hardware accelerators provide the capability to run state-of-the-art DL models with very low power requirements [54].



Acronyms: Inertial Measurement Unit (**IMU**); Database (**DB**); Vision Processing Unit (**VPU**); Onboard Computer (**OBC**); Electronic Power Subsystem (**EPS**); Housekeeping (**HK**)

Figure 2.1: System Context Diagram for vision-based state estimation for landing missions.

This research will focus on the steps from the camera input up to and including the navigation filter. Both instantaneous (i.e. randomised positions above the Lunar surface) performance as well as in an orbit scenario will be analysed to provide enough evidence to warrant further development of the proposed system.

2.1.5. Use Case

The system will be tested in a simulated environment in orbit around the Moon (<500 [km]) in a LIS situation, with attitude information available. The reason why this assumption can be made is because star trackers have become accurate, robust, and inexpensive [70]. Together with an accurate time stamp and SPICE kernels [61], the navigation problem narrows down to finding a correspondence between the camera input and the spacecraft's selenographic position. This is inherently a much more difficult problem to solve than LIS star tracking, since available features (e.g. impact craters) suffer from high dimensionality and varying internal (e.g. viewing angle) as well as external factors (e.g. lighting conditions, surface type, etc.).

This is therefore an excellent problem for an AI-based approach, since this approach has proven to be successful at solving high-dimensional problems [65] like object detection. Using the proposed system in a situation where state information is known is obviously an option as well, and it is expected to further improve accuracy. However, in the interest of designing a fully autonomous system that can initialise its state estimator without ground tracking, this is not considered during the design, development, and verification of the proposed system.

The Moon was chosen as a target because it is a scientifically rich target to land on and because of the large amount of high-quality data available for one to work with. However, this does not mean that the system is limited for use around the Moon only. In fact, the system will be required to function around any significant body that has a cratered surface.

2.1.6. System Objectives

The high level performance objectives can be divided into four categories:

1. **Accuracy**: Provide state estimation filter with accurate absolute state estimations with manageable error from a LIS situation.
2. **Autonomy**: Function without ground station input.
3. **Compatibility**: Interfacing with other (Guidance Navigation & Control (GNC)) components.
4. **Robustness**: Function in a wide variety of environments, including but not limited to different altitudes, viewing angles, surface types, and lighting conditions. This also refers to the system's capability to work with little to no state knowledge available (LIS).

The end-goal of this project is to create a Technology Readiness Level (TRL) 3 [82] software-based demonstrator of a TRN system using region-based object detection and crater pattern matching. Ultimately, the system's purpose is twofold: to demonstrate the performance of AI-based techniques in GNC applications and to provide an open source implementation to the spacecraft avionics community for future comparative testing and development.

2.1.7. Technology Readiness Assessment

Because this system requires hardware components, the system shall be validated up to TRL 3. According to [82]: "*TRL 3 includes both analytical and experimental approaches to proving a particular concept*". The proposed system comprises several software components that require integration before being able to generate the desired results. This means that - in TRL jargon - the software demonstrator is defined as the technology concept, and the analytical and experimental studies to validate hypotheses regarding this technology serve to verify the system at this stage of development. Advancement to TRL 4 would require the software to be written and optimised for an embedded target, which would then be used for further rigorous breadboard testing in a relevant environment.

2.2. Top-Level Requirements

To achieve the end-goal of this project, a set of requirements that cover the main use case (TRN) are generated. The top-level requirements offer a solution to a range of problems related to TRN, taking into account the research domain and questions to maximise the scientific value of this project.

2.2.1. Functional Requirements

FR-01 The system shall be able to deliver absolute position estimations based on attitude information and camera input alone.

Rationale : The main goal of any TRN system is to generate position estimations, the limited available state information during execution is purposely set for the system to function in a LIS situation.

FR-02 The system shall utilise AI techniques.

Rationale : To explore the performance of AI-based algorithms in GNC systems to add to the body of knowledge concerning these types of solutions [29].

FR-03 The system shall be used for navigating around the Moon.

Rationale : To clearly set the scope of this project, the system shall be designed to work for Lunar missions. That is not to say that the technology can be extended to work on other crater-rich targets as well.

FR-04 The system shall comprise all elements required for it to be reproduced by any user that requires it.

Rationale : The system is to be designed with reproducibility in mind. To be able to distribute the solution for further research & development, as well as to encourage reproduction studies for further verification.

FR-05 The system shall be executable entirely on-board a spacecraft.

Rationale : No form of ground tracking is allowed for the purpose of creating an entirely self-contained system. This means that limited resources are available (power, processing, and memory).

FR-07 The system shall function for any Lunar approach path.

Rationale : In order to become a generally applicable system for Lunar landing missions the system must be capable of configuration or self-adaptation to the proposed mission.

2.2.2. Performance Requirements

PR-01 The system shall achieve a maximum absolute state error of less than 487 [m] in a Lunar orbit below 500 [km].

Rationale : This is to solve the challenge of high-precision autonomous navigation where accurate state information in an early stage (higher altitudes) of the approach and descent manoeuvres is desired. The value is derived from the strictest error requirement given in Table 2.1 ($\sqrt{230^2 + 430^2} \approx 487$ [m]).

PR-02 The system shall deliver state updates at a frequency of at least TBD [Hz].

Rationale : To feed the state estimation filter with enough updates to gain sufficient state knowledge confidence. The required update frequency is not known beforehand as it is dependent on the filter's design and the TRN system's performance. Further tests shall have to determine an appropriate rate for the requirement to be validated.

Distance to surface	**Horizontal Error**	**Vertical Error**
200km	1640 [m]	3380 [m]
100km	410 [m]	845 [m]
80km	240 [m]	610 [m]
50km	230 [m]	430 [m]

Table 2.1: Maximum absolute state error requirement for varying altitudes. Values have been extrapolated from [29].

2.2.3. Operational Requirements

OR-01 The system shall be capable of functioning above any cratered region of the Moon without hardware changes.

Rationale : This is to ensure that the system is reusable and is easily integrable with spacecraft. This is inspired by the capability of star trackers to function in the same way, with attitude state as output.

OR-02 The system shall fulfil its task autonomously.

Rationale : Ground operator actions may not be possible at all times, therefore the system shall be designed with autonomy as a primary method of operation.

OR-03 The system shall be a self-contained instrument that is capable of handling all necessary steps to transform optical input and attitude information into absolute state knowledge.

Rationale : It cannot be expected that auxiliary subsystems handle (part of) the work required to execute the system to be designed to deliver absolute state estimations from image input and attitude.

2.2.4. Environmental Requirements

ER-01 The system shall be capable of operating in a lunar orbit environment.

Rationale : The space radiation environment can impact performance or halt execution entirely. Appropriate steps must be considered to minimize this risk.

ER-01.1 The system shall be capable of operating under a Single-Event Effect (SEE) error rate of TBD.

Rationale : A quantification of ER-01 in terms of SEE.

ER-01.2 The system shall be capable of operating with a Total Ionising Dose (TID) of up to TDB Gy.

Rationale : A quantification of ER-01 in terms of TID.

ER-02 The system shall be capable of operating in temperatures ranging from [TBD] to [TBD] [° Celsius].

Rationale : Specifically too high temperatures could limit the cooling capacity available to run the components required to execute the system at an acceptable frequency.

2.3. Component-Level Requirements

Following from the set system-level requirements, an overview of the required functionality per component is given through a functional breakdown diagram. This displays the hierarchical relationship between the top-level requirements and the component-level requirements.

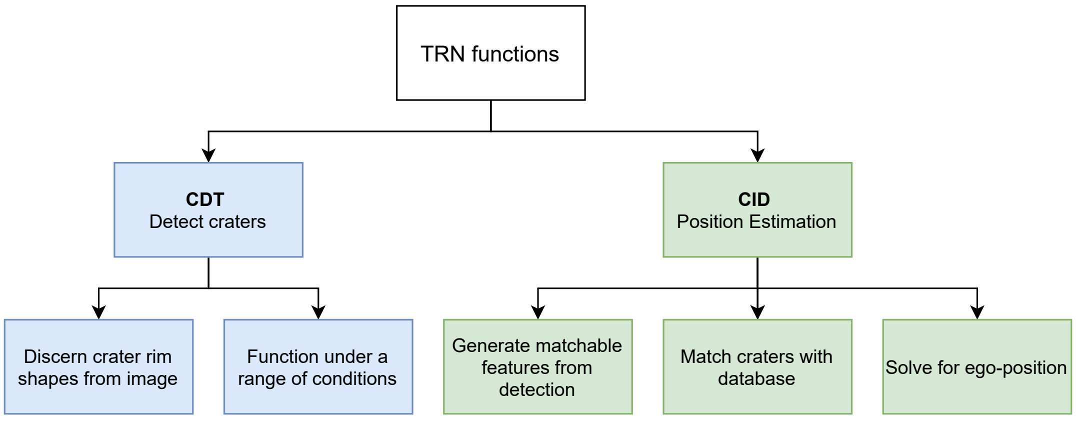

TRN functions
CDT
Detect craters
CID
Position Estimation
Discern crater rim shapes from image
Function under a range of conditions
Generate matchable features from detection
Match craters with database
Solve for ego-position

Figure 2.2: Functional Breakdown Diagram for TRN.

Figure 2.3 displays this sequential relationship between the components.

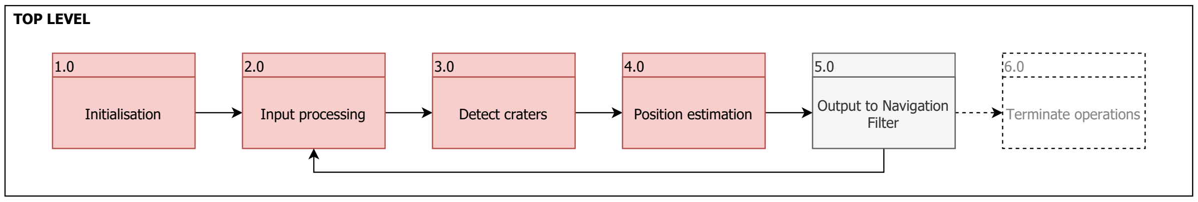


Figure 2.3: Top level functional flow diagram.

2.3.1. Crater Detection Subsystem

Setting clear requirements for the CDA subsystem requires a specification of the desired output for the identification subsystem. It seems clear that, as shown in section 1.4, fitting crater ellipses in the image plane to crater rims yields enough information on individual craters to infer state information. Hence the need for the CDA subsystem to discriminate individual (instanced) crater ellipses from the camera input.

CDT-01 The crater detection subsystem shall be able to detect crater shapes from a camera input.

Rationale: Craters are deemed the most prevalent and easily detectable features present on the Lunar surface.

Functional Requirements

CDT-FR-01.1 The crater detection subsystem shall be able to detect unobscured craters from a position in orbit above the Lunar surface.

Rationale: *The main functionality of a crater detection subsystem.*

CDT-FR-01.2 The crater detection subsystem shall be able to discern individual instanced crater rims.

Rationale: *To be able to detect individual instances of craters as opposed to merely per-pixel classification of the input image like DeepMoon [78].*

CDT-FR-01.3 The crater detection subsystem shall be able to convert a detected crater rim into parameterised ellipse values.

Rationale: *This is the output that is transmitted to the crater identification & position estimation subsystem.*

Performance Requirements

CDT-PR-01.1 The crater detection subsystem shall be able to detect craters with attitudes of up to 30 [degrees] away from nadir-pointing attitude.

Rationale: *To prove that the added flexibility of using off-nadir pointing works as expected in the results given in [17] (p.76).*

CDT-PR-01.2 The crater detection subsystem shall be able to detect craters in sunlit conditions with varying sun angles.

Rationale: *This is so that the system can be deployed with varying sun angles, increasing the system's flexibility compared to LunaNet [24] which only augmented images by adjusting the brightness.*

CDT-PR-01.4 The crater detection subsystem shall be capable of detecting at least 3 craters per given camera input.

Rationale: *Crater pattern matching works by generating features per sets of craters, usually more than 3 [17] to be discriminating enough.*

CDT-PR-01.5 The crater detection subsystem shall be able to detect craters with maximum ellipse axis error of 3 [pixels].

Rationale: *Accuracy is desired for accurate ego-position estimation, and from [17]: "we observe that matching performance is not appreciably affected by ellipse localisation error until these errors reach about 2–3 pixels" (p. 76).*

CDT-PR-01.6 The crater detection subsystem shall be able to detect craters with maximum ellipse location error of 3 [pixels].

Rationale: *Similar to CDT-PR-01.5, also [17] mentions the fact that the "crater localisation error needs to be better than 2-3 pixels" (p. 77).*

CDT-02 The crater detection subsystem shall be able to detect craters at a sufficient rate so as to fulfil **PR-02**.

Rationale: *Discerning crater rims from the sensor input can be relatively time consuming depending on the methodology, hence it is necessary for this subsystem to operate at a high enough throughput to keep the identification subsystem saturated. This requirement and all its sub-requirements will have to be verified in a TRL 4 demonstrator, but needs to be defined at this stage to ensure future compatibility. [6].*

Functional Requirements

CDT-FR-02.1 The crater detection subsystem shall utilise a hardware accelerator.

Rationale: *DL models cannot be run on space-grade hardware with realistic throughput [53]. Furthermore, this system is a use case [29] for mobile hardware accelerators such as the Intel-Movidius Myriad line of chips that are currently seeing adoption in space platforms*

Performance Requirements

CDT-PR-02.1 The crater detection subsystem shall be able to process input images at a rate of TBD [Hz].

Rationale: The quantification of CDT-02, which requires the crater detector to strike a balance between fidelity and speed. It is yet to be determined what the required update rate is for a complete TRN system with state estimation filter included.

CDT-PR-02.2 The crater detection subsystem shall be able to process input images with a resolution of at least TDB [pixels].

Rationale: Follows from CDT-PR-02.1: resolution is one of the primary factors that influence the throughput of a dedicated hardware accelerator.

CDT-03 The crater detection subsystem shall be able to interface directly with the crater identification subsystem.

Rationale: Efficient communication between both the detection and the identification subsystem is necessary to ensure successful state estimation.

Functional Requirements

CDT-FR-03.1 The crater detection subsystem shall be able to generate a set of detected crater rims which comprises parameterised ellipses describing the crater rims in the image plane.

Rationale: This is required for crater pattern matching in general.

CDT-FR-03.2 The crater detection subsystem shall be able to communicate its state to the crater identification subsystem and the GNC subsystem.

Rationale: Required for appropriately handling errors or other exceptional events.

CDT-FR-03.3 The crater detection subsystem shall be able to communicate detection confidence values to the matching subsystem.

Rationale: This can possibly be used to generate position estimation confidence values that can be communicated to the state estimation subsystem.

The top-level requirement **CDT-01** without its sub-requirements could at first glance be achievable using the solutions given in section 1.2. However, the sub-requirements that have been introduced highlight the shortcomings of these solutions. For example: they have not been designed to work using optical input (DeepMoon [78]), or detect instanced craters without the need for sensitive post-processing steps (LunaNet [24]). **CDT-02** is also strongly related to the CDA architecture, since it can influence the degree with which it can run on a hardware accelerator, keeping in mind **FR-01** and the system context presented in Figure 2.1.

The functional flow is therefore defined as:

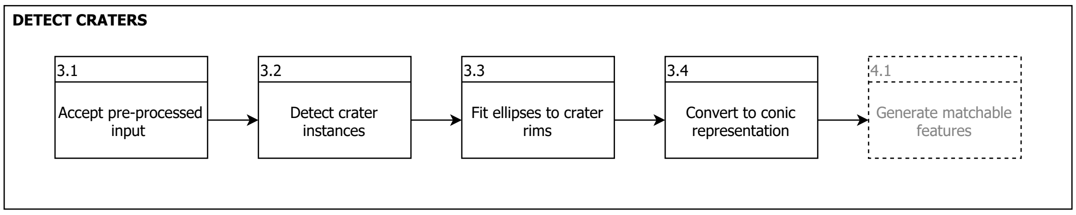

DETECT CRATERS
3.1
Accept pre-processed input
3.2
Detect crater instances
3.3
Fit ellipses to crater rims
3.4
Convert to conic representation
4.1
Generate matchable features

Figure 2.4: Crater detection functional flow diagram.

2.3.2. Absolute Position Estimation Subsystem

CID-01 The position estimation subsystem shall be able to match detected crater shapes with a database.

Rationale : In order to be able to derive ego-position estimations, it is necessary to match detected features with a database from which the craters' locations can be retrieved.

Functional Requirements

CID-FR-01.1 The system shall be able to match triads of approximately coplanar craters.

Rationale: From the theoretical approach described in [17] for orbit situation below 500 [km].

CID-FR-01.3 The system shall be able to match detected craters from varying altitudes of at most 500 [km] above the Lunar surface.

Rationale: Altitude invariance is key when applying the system in a Lunar approach scenario.

CID-FR-01.4 The system shall be able to match detected craters with off-nadir pointing camera angles.

Rationale: To prevent the system constraining allowed attitudes during manoeuvres.

CID-FR-01.5 The system shall be able to generate a database specific to a desired mission profile.

Rationale: Derived from FR-04, and allows for fine tuning the system for a specific region above the Lunar surface.

CID-FR-01.6 The system shall be able to match craters from noisy measurements.

Rationale: Essentially caused by the non-linear and unstructured nature of the input data. This is one of the key aspects in which a AI-based system can outperform traditional CV methods.

Performance Requirements

CID-PR-01.2 The system shall be able to create a database for a cratered region that spans at least 1000 by 1000 [km].

Rationale: A sufficiently large area is set for verifying the results shown in [17].

CID-02 The system shall be able to derive the spacecraft's ego-position from the set of identified craters and the spacecraft attitude.

Rationale : The output of this system is the spacecraft's ego-position, therefore a final post-processing step needs to be performed to reach this result.

Functional Requirements

CID-FR-02.1 The system shall be capable of deriving ego-position estimations from identified craters and spacecraft attitude alone.

Rationale: Essentially the main functionality of the subsystem, with the assumption on spacecraft attitude knowledge derived from [17]: "... we can think of no plausible failure mode (where recovery is still possible) in which the spacecraft has no knowledge of time (necessary for finding lunar attitude from SPICE kernels) or of inertial attitude".

CID-FR-02.2 The system shall be capable of rejecting faulty position estimates.

Rationale: It is assumed that the system will return erroneous position estimations at a to-be-determined rate, which need to be handled appropriately.

CID-FR-02.3 The system shall be capable of integrating prior state information should it be available to improve accuracy.

Rationale: A sufficiently large area is set for verifying the results shown in [17].

Performance Requirements

CID-PR-02.1 The system shall be able to derive instantaneous ego-position estimations with an average accuracy below 1 [km].

Rationale: See CID-PR-02.3.

CID-PR-02.3 The system shall be able to derive ego-position estimation error below 500 [m] when multiple measurements are combined through state estimation filtering.

Rationale: See PR-01, and consider that noisy instantaneous ego-position estimations can be used in a filter (e.g. an EKF) to gain more accurate state information. It is hypothesised that if a state estimation error below 500 [m] can be achieved, the LIS situation is alleviated. This would allow methods requiring approximate state information and relative position updates to be used to converge further.

CID-PR-02.4 The system shall be able to function with an initial state uncertainty of at most 500 [km].

Rationale: To verify the system's performance in a LIS scenario.

CID-03 The system shall be able to identify craters and derive the spacecraft's ego-position at a sufficient rate so as to fulfil **PR-02**.

Rationale : *Similar to CDT-02, matching crater features from the sensor input can be relatively time consuming depending on the methodology, hence it is necessary for this subsystem to operate time efficiently to keep the identification subsystem saturated. This requirement is mostly aimed at a TRL 4 engineering model, and is kept open as more knowledge is available about an embedded version of the proposed system.*

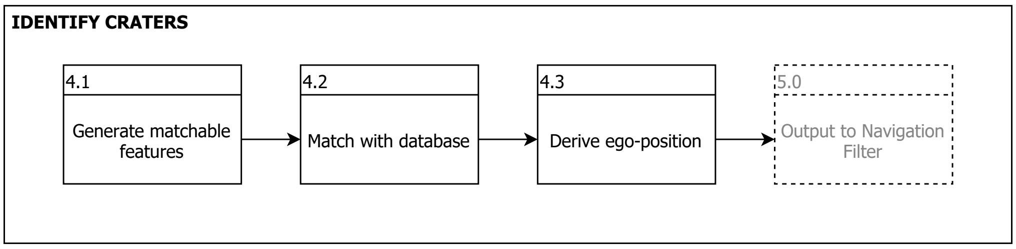


Figure 2.5: Crater identification functional flow diagram.

2.3.3. System Block Diagram

Now that it has been defined what each component must achieve individually, an overview that describes the system at a more technical level is needed. Using the context diagram (Figure 2.1) as a basis and using the gained background knowledge for TRN, a block diagram showing the in- and outputs of each component and the state estimation loop that it is part of.

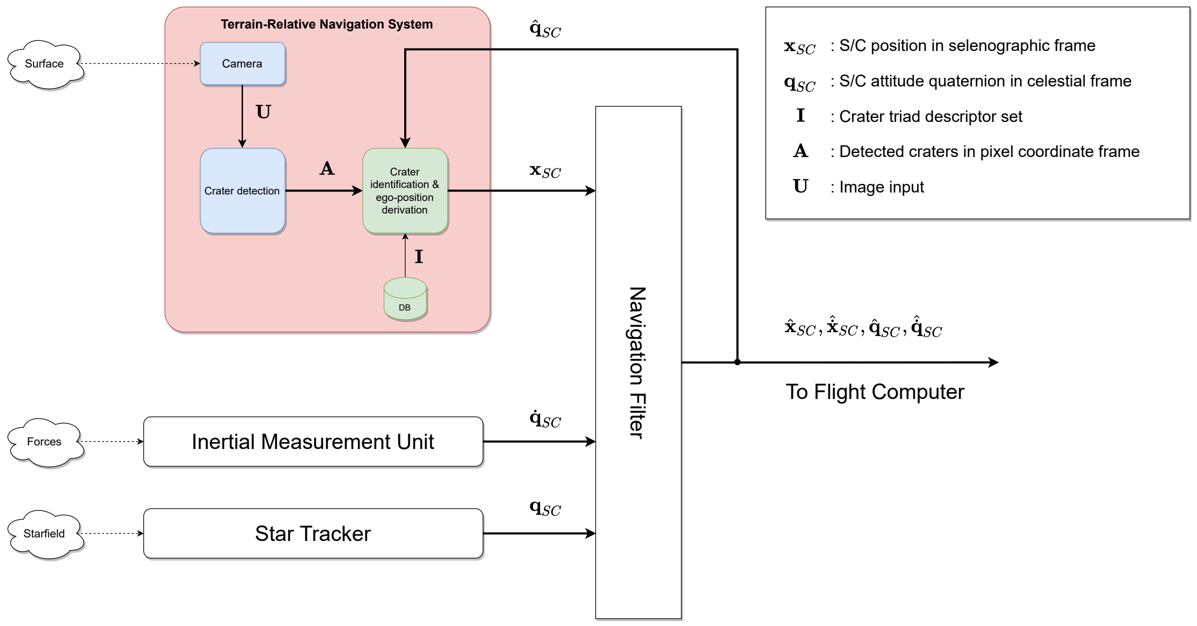

Terrain-Relative Navigation System
Surface
Camera
$\mathbf{U}$
Crater detection
$\mathbf{A}$
Crater identification & ego-position derivation
$\mathbf{I}$
DB
$\hat{\mathbf{q}}_{SC}$
$\mathbf{x}_{SC}$
Navigation Filter
$\hat{\mathbf{x}}_{SC}, \hat{\dot{\mathbf{x}}}_{SC}, \hat{\mathbf{q}}_{SC}, \hat{\dot{\mathbf{q}}}_{SC}$
To Flight Computer
Forces
Inertial Measurement Unit
$\dot{\mathbf{q}}_{SC}$
Starfield
Star Tracker
$\mathbf{q}_{SC}$
$\mathbf{x}_{SC}$: S/C position in selenographic frame
$\mathbf{q}_{SC}$: S/C attitude quaternion in celestial frame
$\mathbf{I}$: Crater triad descriptor set
$\mathbf{A}$: Detected craters in pixel coordinate frame
$\mathbf{U}$: Image input

Figure 2.6: System Block Diagram for the proposed Terrain-Relative Navigation (TRN) system.

2.4. Requirements Verification & Validation Plan

Validation of the full system cannot be performed within the confines of this research, as it requires the design to progress to higher TRL levels. Since the goal is a TRL 3 demonstrator and therefore not a physical engineering model, the system cannot be tested in a representative environment using a physical test bench. For this reason, validating the demonstrator is answering the question of whether the proposed system merits advancement to TRL 4, and in such a way attempting to find out if the concept functions in a representative simulation environment. Assessing whether the proposed technology can be considered TRL 3 requires the description, requirements, verification, and viability to be assessed [82].

By first verifying the demonstrator by analytically and experimentally, a strong case can be made to invest in a more mature version of the system for further testing. The addition to the body of knowledge concerning AI-based TRN is a similarly important aspect of this research: the combination of a region-based object detection CDA and a projective-invariant-based crater patter matching algorithm is a novel development in the field of TRN. Furthermore, general usage of AI techniques in space applications is a very active field of research.

Table 7.1 displays the type of tests that are performed to verify the requirements set for the proposed TRN system. Verified performance requirements imply that the system also complies with functional requirements. Requirements that need verification through the use of an engineering model of the system (TRL 4) are deemed out of scope, but are included for integrity. A simulation approach is taken to design, develop, and verify the functional and performance requirements of the subsystems and eventual end-to-end system. This requires the development of a robust testing environment that can generate representative scenes. Three different test types are defined to assess the functionality and performance of the proposed system: pseudo-measurements, instantaneous camera simulation tests, and trajectory simulation tests.

Pseudo-measurements are used to develop and test the crater identification and position estimation subsystem. Essentially, these consist of projected catalogue craters that have been artificially perturbed. These pseudo-measurements will then be used as input, which also means that the perturbation that is used can be added in a controlled manner to assess the system's sensitivity to noise.

Instantaneous camera simulation tests require the use of a representative scene generator which can emulate the camera output for a given position and attitude above the Lunar surface. Using this, a series of Monte Carlo trials where the position and attitude of the camera can be drawn from a set of pre-defined distributions can be performed. It is assumed that because of the utterly random nature of the input image given a random state above the Lunar surface that the system can be tested with little to no bias introduced by the test data. Developing a test suite that can perform the aforementioned is a novelty, and has the potential to aid in further development beyond the scope of this project.

Finally, to validate whether the demonstrator can become the basis for a system that can fulfil the demand of "*highly accurate relative navigation in a non-linear and unpredictable harsh environment*" a representative test is performed in which the system is fed with simulated input of a camera while in orbit. This means that multiple measurements through time can be used to generate more accurate state information through the use of an Extended Kalman Filter (EKF). This test mainly serves to validate the proposed system's functionality in a representative environment, and should indicate whether the demonstrator merits advancing the technology to the next TRL.

3 Data Generation

Designing and developing the systems for detecting and identifying crater features requires a robust method of generating realistic testing data. This split into two categories: input and target data. The former can be described as simulating the camera input as seen in Figure 2.1 and Figure 2.6 as accurately as possible. The target data is a broader term, as it needs to contain detection targets, as well as the spacecraft state to be able to design the crater detector and identifier, respectively. To achieve this, an object-oriented data generator was designed as this holds significant benefits when translating the concepts to software.

3.1. Defining Orbiting Body

The core problem that is being solved in this project is estimating the body state, hence the need for a solid and extendable base class for a generic orbiting body.

3.1.1. Reference Frames

The absolute state is defined in a Moon-Centred-Moon-Fixed (MCMF) reference frame, as denoted by $\mathscr{M}$. The body's attitude is always initialised with a nadir-pointing attitude defined by Equation 3.1, Equation 3.2, and Equation 3.3, which defines an East-South-Down coordinate system for X, Y, and Z respectively. The reasoning for this is that it follow standard convention for camera reference frames where the Z-axis is the Line-of-Sight (LoS) vector, the Y-axis is pointing downwards from the top-left corner of the image plane, and the X-axis is pointing rightwards in the image plane.

$$\boldsymbol{d}_B = \frac{-\boldsymbol{x}_B}{||\boldsymbol{x}_B||}, \tag{3.1}$$

$$\boldsymbol{e}_B = \frac{\boldsymbol{k} \times \boldsymbol{x}_B}{||\boldsymbol{k} \times \boldsymbol{x}_B||}, \tag{3.2}$$

$$\boldsymbol{s}_B = \frac{\boldsymbol{d}_B \times \boldsymbol{e}_B}{||\boldsymbol{d}_B \times \boldsymbol{e}_B||}, \tag{3.3}$$

with

$$\boldsymbol{k} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T, \tag{3.4}$$

yields the following transformation matrix for the body-centred reference frame ($\mathscr{B}$) w.r.t. MCMF reference frame:

$$\boldsymbol{T}^{\mathscr{M}}_{\mathscr{B}} = \begin{bmatrix} \boldsymbol{e}_B & \boldsymbol{s}_B & \boldsymbol{d}_B \end{bmatrix} \tag{3.5}$$

This parameter will henceforth be referred to as $\boldsymbol{T}_B$. Subsequent attitude changes are enacted through methods that either incrementally change the attitude or manually set the attitude by either inputting a new transformation matrix (from $\mathscr{M} \rightarrow \mathscr{B}$) or quaternion.

3.2. Defining Camera Properties

Next, the camera system that is used to view the Lunar surface to identify craters is defined by its projective properties. A pinhole camera model is considered sufficient in the context of the problem since the same methodology should hold for more complex definitions of $\boldsymbol{K}$. Subsequent iterations of the concept proposed in this paper can adopt a more complex sensing model, which should have little effect on the overall performance as long as the parameters are measured accurately.

As mentioned before, the coordinate system is defined in a Z-forward setup, meaning the X- and Y-coordinates are reserved for the 2D image plane.

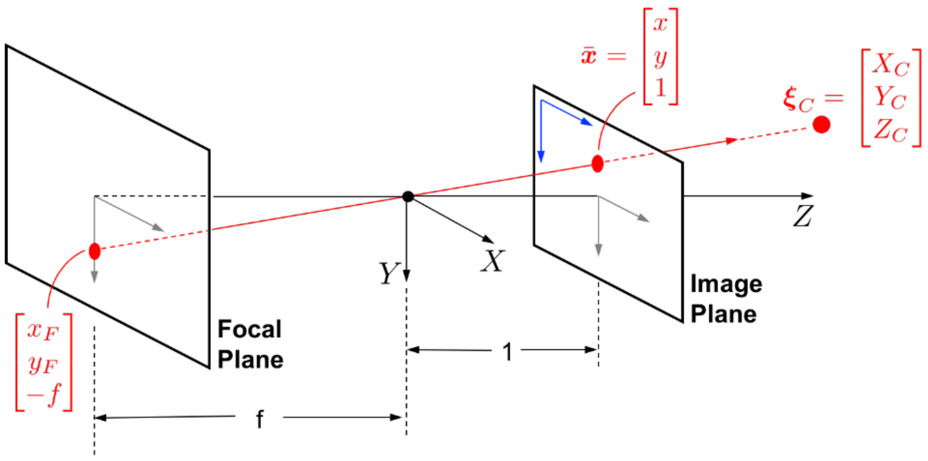

$\bar{\boldsymbol{x}} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
$\xi_C = \begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix}$
Z
X
Y
Image Plane
Focal Plane
1
f
$\begin{bmatrix} x_F \\ y_F \\ -f \end{bmatrix}$

Figure 3.1: Pinhole camera model illustration, reprinted from [16].

$$\boldsymbol{K} = \begin{bmatrix} d_x & \alpha & u_p \\ 0 & d_y & v_p \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.6}$$

or written more compactly:

$$\bar{\boldsymbol{u}} = \boldsymbol{K}\bar{\boldsymbol{x}} \tag{3.7}$$

where $\bar{\boldsymbol{x}} = \begin{bmatrix} x & y & 1 \end{bmatrix}^T$ and $\bar{\boldsymbol{u}} = \begin{bmatrix} u & v & 1 \end{bmatrix}^T$.
Relating the camera matrix to its cartesian position yields the projection matrix:

$$\boldsymbol{P} = \boldsymbol{K}\begin{bmatrix} \boldsymbol{T}_B & -\boldsymbol{x}_B \end{bmatrix} \tag{3.8}$$

Using the inherited positional and rotational state from the orbiting body class therefore completes the camera class, being able to relate points or basic shapes in the MCMF coordinate system to the image plane.

3.3. Simulating Physically Accurate Scenes

One of the proposed system's main benefits is its flexibility, meaning it has the potential to be deployed over a wide range of mission scenarios with a range of lighting and surface conditions, as well as varying altitude and orientation. Simulating scenes that can accurately reflect real-life situations in low-orbit missions around the Moon is therefore done by employing scientifically accurate rendering software that is tuned for space missions: SurRender [67]. This software is marketed as *"Image generation software for space exploration"*[1], and is capable of using DEM data and associated textures to render synthetic camera data. This is well-suited to the fulfilment of the set functional requirements, as it enables virtually limitless data generation for development and testing purposes.

[1]SurRender about page [link]

3.3.1. Data

At the core of this scene lies the physical data describing the Moon shape and colour, which has been created using the Planetary Data System (PDS) node DEM based on data that was captured using NASA's Lunar Orbiter Laser Altimeter (LOLA) [81] on board the Lunar Reconnaissance Orbiter (LRO) [83]. This dataset was generated by combining over 6.5 billion measurements made between 2009 and 2013, and has an average accuracy of approximately 20 and 1 meters horizontally and radially, respectively. This was then converted into an image with a resolution of 256 pixels per degree, resulting in 118 meter wide pixels at the equator. There are gaps present in the data that range between 1 and 4 km wide which have been filled up by interpolation [7]. See Figure 3.2 for an overview of the DEM that is used for this project. More recent data exists that exhibit a higher pixel density, such as the result of the merge of data from both LOLA and SELENE Kaguya launched by Japan Aerospace Exploration Agency (JAXA) [9]. However, this data only ranged between latitudes 60.0°N and 60.0°S, effectively removing both Lunar polar regions from the simulation.

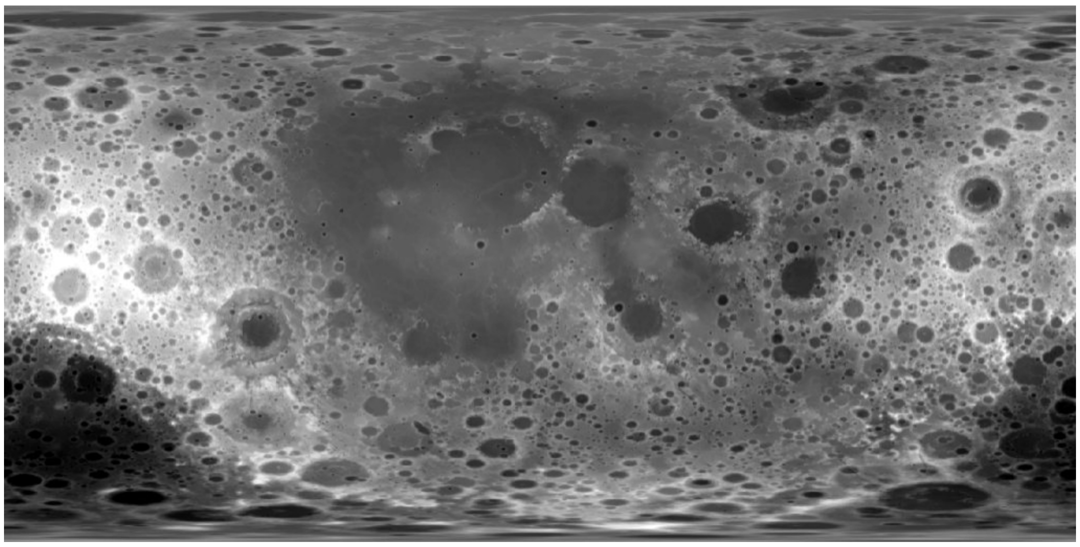


Figure 3.2: Low resolution version of Moon LRO LOLA DEM, reprinted from [7]. Pixel values indicate height w.r.t. mean Moon radius.

Furthermore, NASA's SPICE kernels [61] were used to determine accurate relative positions of the Sun and Earth to ensure both lighting angles that represent the real world, as well as the flexibility to generate data for specific moments in time. In fact, this flexibility is utilised in the generation of large datasets where the "scene time" is dynamically changed to ensure a sunlit surface downrange from the camera.

3.3.2. Results

A straight forward method of verifying the functionality of both the renderer and the classes upon which it is based, is by visually inspecting the output of a the renderer above a well-known crater such as Tycho [5]. This has the benefit of being able to eliminate many possible issues that may arise in the pipeline required to simulate the output such as unexpected visual artifacts, mismatching locations, or unrealistic lighting. Quick inspection of Figure 3.3 verifies that the pipeline is functioning correctly. What is especially interesting is how well the shadows match the real image, which is ultimately going to effect how well the detection model can function in real-world situations under varying lighting situations. The output image is grayscale and is scaled to have pixel values between 0 and 1.

Combining all attributes and methods from both the orbiting body and camera class and creating an interface with the SurRender renderer yields a flexible generator that can be used to generate highly configurable synthetic camera output. However, the class has been defined in way that it can be used with other rendering software that have Python interfaces as well.

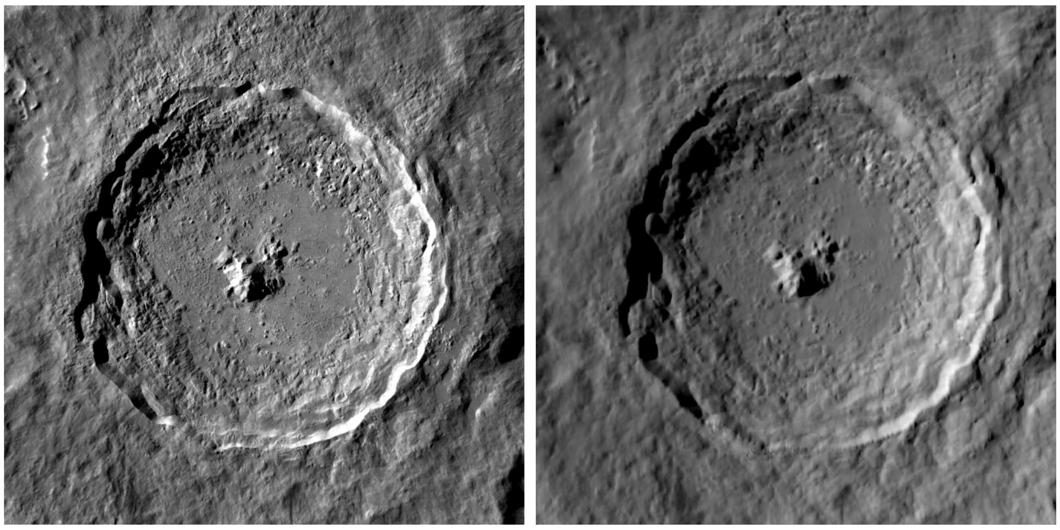


Figure 3.3: Comparison of real image shot of the Tycho crater on the Moon captured by NASA's LRO camera module [2] (left), and an image generated using SurRender combined with the custom interface as outlined in Figure 3.8 (right). The renderer was initialised above selenographic coordinates 43.31°S 11.36°W at an altitude of 140 km with a solar incidence angle of approximately 70°.

3.4. Projecting Craters

Crater databases are often structured in a tabular structure containing positional information (latitude, longitude) and geometric information. In [71] it is argued that although crater shapes have been approximated as circles before [46, 76], it is deemed more accurate to describe them as ellipses, a generalisation of a circle. The database that was used for this project is described in [71] and contains craters >1-2 km, and was generated using elevation maps of the Moon. Parameters such as the a and b, as well as the rotation w.r.t. the East-West direction are given. The goal is to use these parameters to generate a mask that traces the apparent shapes of the craters in the image plane.

To achieve this, the methodology described in [17] is followed. The craters are defined as conics in a three-dimensional selenographic reference frame through their location and ellipse parameters, using this information combined with the camera's position and characteristics yields a set of projected two-dimensional conics in the image plane.

3.4.1. Crater Position, Orientation, and Shape

First, the two dimensional plane in which a crater resides is defined by its position and relative orientation w.r.t. the MCMF reference frame. The position is determined by its latitude φ_i and longitude λ_i and the Moon radius. In this case the Moon shape is assumed to be spherical because of SurRender limitations, meaning the Cartesian coordinates for each crater is given by:

$$\boldsymbol{x}^{(c)}_{\mathscr{M}_i} = \rho_i \begin{bmatrix} \cos\varphi_i \cos\lambda_i \\ \cos\varphi_i \sin\lambda_i \\ \sin\varphi_i \end{bmatrix} \tag{3.9}$$

The plane's relative orientation is given by a local East-North-Up coordinate system with its origin at the crater centre. Figure 3.4 illustrates the setup in the $\mathscr{M}$ reference frame. The normalised axes are given in Equation 3.10 to Equation 3.12, with Equation 3.13 combining them into a transformation matrix from the $\mathscr{M}$ frame to the crater plane $\mathscr{C}$.

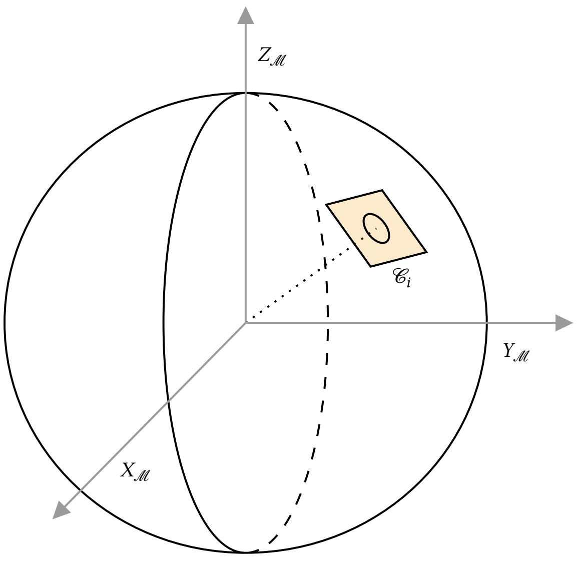


Figure 3.4: Illustration of two dimensional crater plane definition w.r.t. MCMF reference frame.

$$\boldsymbol{u}_i = \frac{\boldsymbol{x}^{(c)}_{\mathscr{M}_i}}{||\boldsymbol{x}^{(c)}_{\mathscr{M}_i}||}, \tag{3.10}$$

$$\boldsymbol{e}_i = \frac{\boldsymbol{k} \times \boldsymbol{u}_i}{||\boldsymbol{k} \times \boldsymbol{u}_i||}, \tag{3.11}$$

$$\boldsymbol{n}_i = \frac{\boldsymbol{u}_i \times \boldsymbol{e}_i}{||\boldsymbol{u}_i \times \boldsymbol{e}_i||}, \tag{3.12}$$

$$T^{\mathscr{C}_i}_{\mathscr{M}} = \begin{bmatrix} \boldsymbol{e}_i & \boldsymbol{n}_i & \boldsymbol{u}_i \end{bmatrix} \tag{3.13}$$

Crater geometry is approximated by fitting an ellipse on the crater rim in $\mathscr{C}$. An ellipse is an example of a conic section, which is a set of points that satisfy the following second-degree polynomial equation:

$$Q(x, y) = Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0, \tag{3.14}$$

with its more compact form:

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} A & B/2 & D/2 \\ B/2 & C & E/2 \\ D/2 & E/2 & F \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \bar{\boldsymbol{x}}^T \boldsymbol{A}_Q \bar{\boldsymbol{x}} = 0, \tag{3.15}$$

meaning that the ellipse can be parameterised as

$$\boldsymbol{A}_Q = \begin{bmatrix} A & B/2 & D/2 \\ B/2 & C & E/2 \\ D/2 & E/2 & F \end{bmatrix} \tag{3.16}$$

Equation 3.16 represents the format in which the crater shape is saved as two-dimensional conic, along with its positional information $\boldsymbol{x}^{(c)}_{\mathscr{M}_i}$. Combined they contain all the information required to know the position in the image plane, provided the camera state is known.

3.4.2. Crater-Camera Homography

Creating the correspondence between catalogued craters in their respective coordinate systems and the image plane is done using the following methodology [17]:

$$H_{\mathscr{M}_i} = \begin{bmatrix} T^{\mathscr{C}_i}_{\mathscr{M}} \boldsymbol{S} & \boldsymbol{x}^{(c)}_{\mathscr{M}_i}, \end{bmatrix} \tag{3.17}$$

with

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{I}_{2\times2} \\ \boldsymbol{0}_{1\times2} \end{bmatrix} \tag{3.18}$$

The conversion between image plane coordinates ($\bar{\boldsymbol{x}}_C$) and digital image pixel coordinates ($\bar{\boldsymbol{u}}_C$) uses the projection matrix given in Equation 3.8, and is described the following affine transformation [17]:

$$\boldsymbol{H}_{\mathscr{C}_i} = \boldsymbol{P} \begin{bmatrix} H_{\mathscr{M}_i} \\ k^T \end{bmatrix} \tag{3.19}$$

The outcome is $\boldsymbol{A_i}$, the projected conic section of the crater in the image plane:

$$\boldsymbol{A_i} \propto \boldsymbol{H}_{\mathscr{C}_i}^{-T} \boldsymbol{C}_i \boldsymbol{H}_{\mathscr{C}_i}^{-1} \tag{3.20}$$

This shape is dependent on both the crater catalogue values, as well as the camera position and characteristics as described in the preceding sections. Because of this dependence, the object-oriented approach proves valuable as it provides an intuitive method of bringing together data and methods to form meaningful results.

3.4.3. Results

To ensure the projection works, a series of demonstration images have been generated to visually inspect the accuracy of the pipeline. Figure 3.5 displays that the apparent positions of the ellipses project correctly on the spherical surface of the Moon. Closer examination shows that many well-pronounced craters are accompanied by a fitted ellipse. Certainly not all craters appear to have an ellipse fitted to them, which is caused by the filter based on crater size and ellipticity imposed on the database to prevent an overcrowded image.

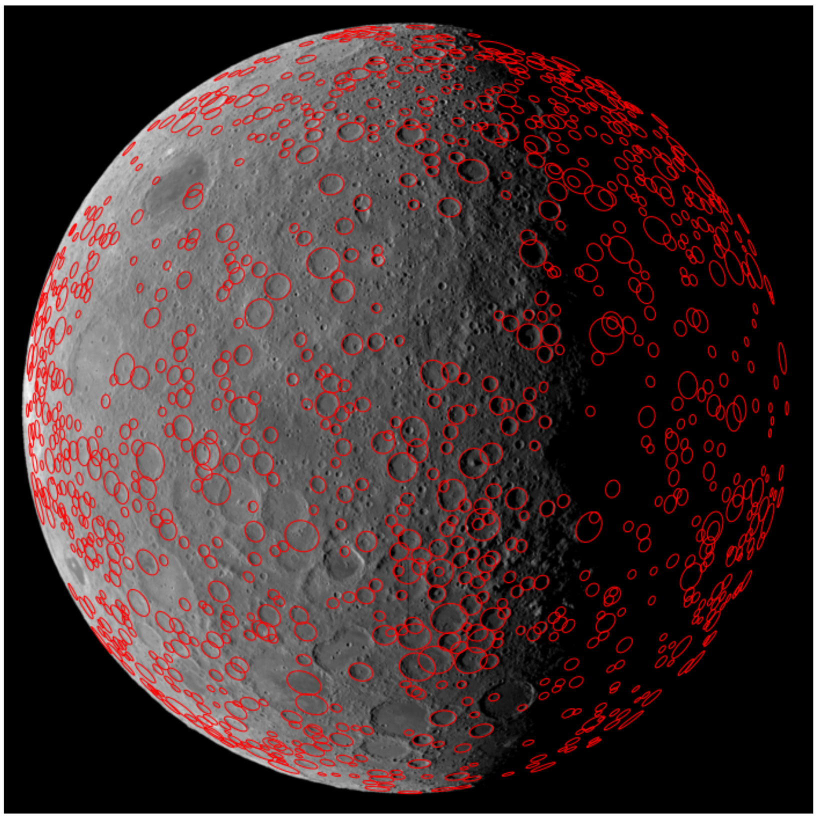


Figure 3.5: Example of projected ellipses overlaid onto a scene of the Moon. Craters with a major axis between 15 and 150 km and a maximum ellipticity of 1.5 were used in the creation of the overlay. The renderer was initialised above selenographic coordinates 0.0°N 180.0°E at an altitude of 3000 km with a solar incidence angle of approximately 72°.

Using the projected ellipses, a pixel-wise mask can be generated to indicate the crater-rims in the image plane. This will prove to be useful when developing the detection algorithm, where it can serve as a ground truth. Figure 3.6 displays an example of one synthetic image-mask pair. The pixel values are set to unique values to facilitate the retrieval of individual crater instances.

The inconsistency of crater rims being present in the mask is something to be aware of, as this may incorrectly disqualify detections during the training phase of the detector. Furthermore, the pipeline also allows the user to define whether to generate filled or outlined ellipses, as well as it being able to discriminate between separate instances of craters rather than just returning a binary mask.

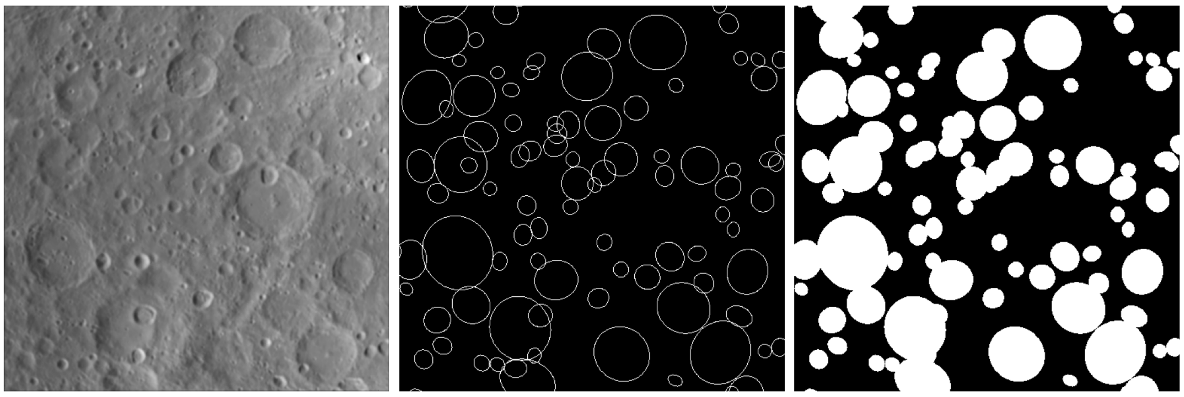


Figure 3.6: Example of an image-mask pair. The renderer was initialised above selenographic coordinates 0.0°N 160.0°E at an altitude of 3000 km with a solar incidence angle of approximately 52°.

3.5. Data Generation Pipeline

The data generation pipeline is defined by the sum of its parts, as can be seen in Figure 3.8. Combining the data present in the Lunar surface DEM [7] and the crater database given in [71] and passing them through a render pipeline using SurRender as well as projecting the given craters allows for flexible dataset creation. Object-oriented design aided in separating the concerns, as well as allowing for separate verification of the modules.

3.5.1. Results

The dataset generation is defined by a loop that randomly generates positions above the Lunar surface using values drawn from the distributions seen in Table 3.1. In addition, for every generated position the scene time and spacecraft attitude are set randomly (while ensuring enough visibility of the Lunar surface). The current setup can generate a dataset of around 80 thousand images at 256 by 256 pixels along with the ground truth masks in about 1 hour on a workstation with the specification given in Table A.1. The ability to generate virtually limitless amounts of data has the benefit of ensuring that the detection model generalises[2] well and prevents over-fitting.

Name	**Symbol**	**Unit**	**Distribution**
Latitude	φ	°	$\mathcal{U}(-90, 90)$
Longitude	λ	°	$\mathcal{U}(-180, 180)$
Altitude	h	km	$\mathcal{U}(150, 140)$
Roll	ϕ	°	$\mathcal{U}(-180, 180)$
Pitch	θ	°	$\mathcal{U}(-30, 30)$
Yaw	ψ	°	$\mathcal{U}(-30, 30)$

Table 3.1: Distributions used for dataset generation. Roll, pitch, and yaw are deviations away from the nadir-pointing vector and are applied in this order.

[2]A model's ability to predict correctly on previously unseen data drawn from the same distribution - in this case images of the lunar surface from a low orbit position [Link]

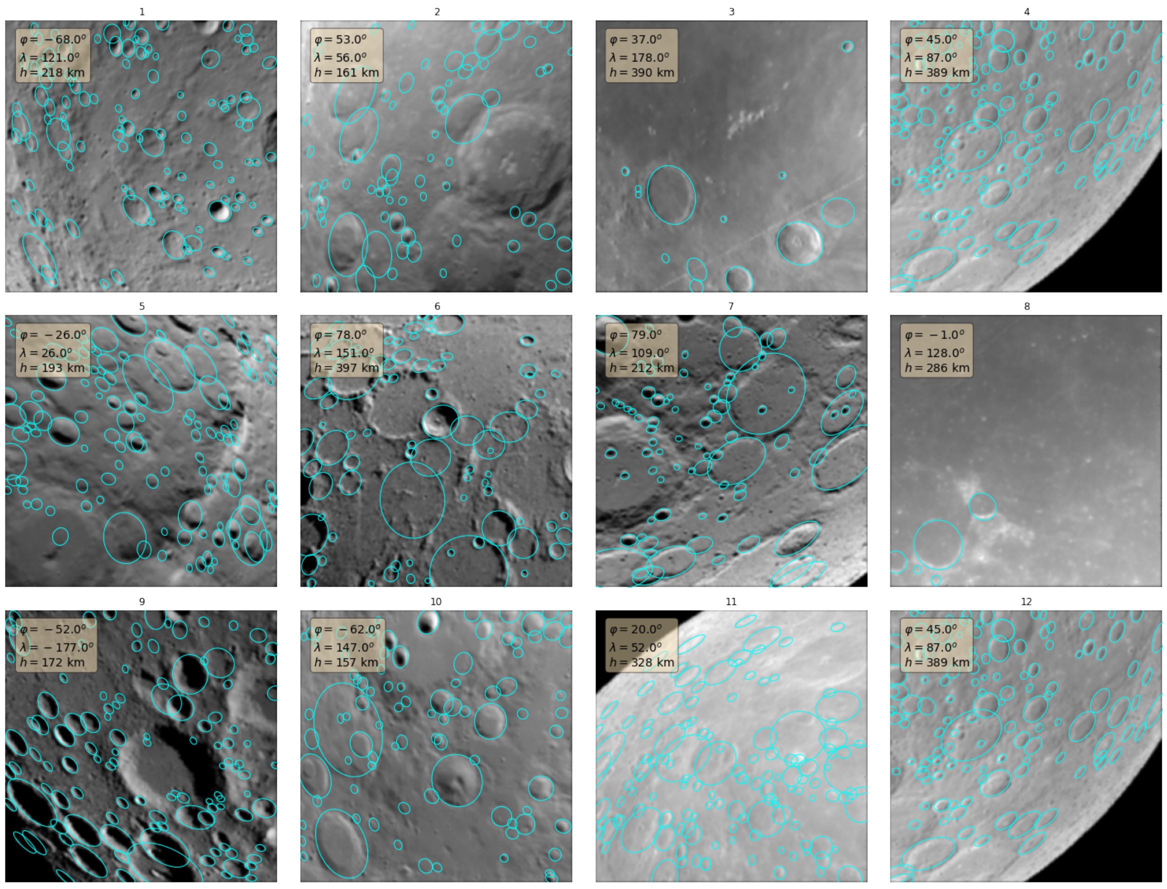


Figure 3.7: Twelve randomly generated synthetic images with the ground truth craters projected onto them. The associated camera positions are given as latitude φ, longitude λ, and height h.

Figure 3.7 displays the variety of the scenes that can be generated using the data generation pipeline. Stark differences between the number of craters, lighting conditions, spacecraft attitude, and surface types can be seen in this small dataset alone. Lighting variation seems to have a great impact on the visible features of a crater, comparing image #11 with #9 shows this difference. Being able to discern craters by eye actually serves as a good indication on whether a crater detection algorithm can determine a crater shape from pixel values. This demonstrates the vulnerability of using a passive sensor for crater detection, as the features that generally define a crater in an image are heavily related to the sun angle and therefore shadow it casts. Besides this, it needs to be stated that while most projected craters seem like a good fit for the shape of a crater, some instances such as the bottom left crater in image #12 appear maligned in the image plane. This can be caused by a variety of reasons ranging from the method that was used to create the crater database [71], to the way that SurRender handles altitude deviation with respect to the Moon radius. Nevertheless, the amount of correctly fitted craters vastly outnumber the erroneous ones, meaning that meaningful features can be derived for training the detector.

Having access to a large dataset containing datapoints with synthetic images, apparent crater shapes, and relevant camera state information (position, attitude) enables further research into more robust CDAs. Hence why this pipeline is considered one of the main additions to the body of knowledge concerning CDA and TRN in general. This also further motivates the choice of programming language, Python, as it is the main language that allows researchers to build AI models. Having the output be available in native NumPy arrays allows for fast prototype creation, which undoubtedly is one of the main critical steps in the process towards developing a new solution for the posed navigation problem.

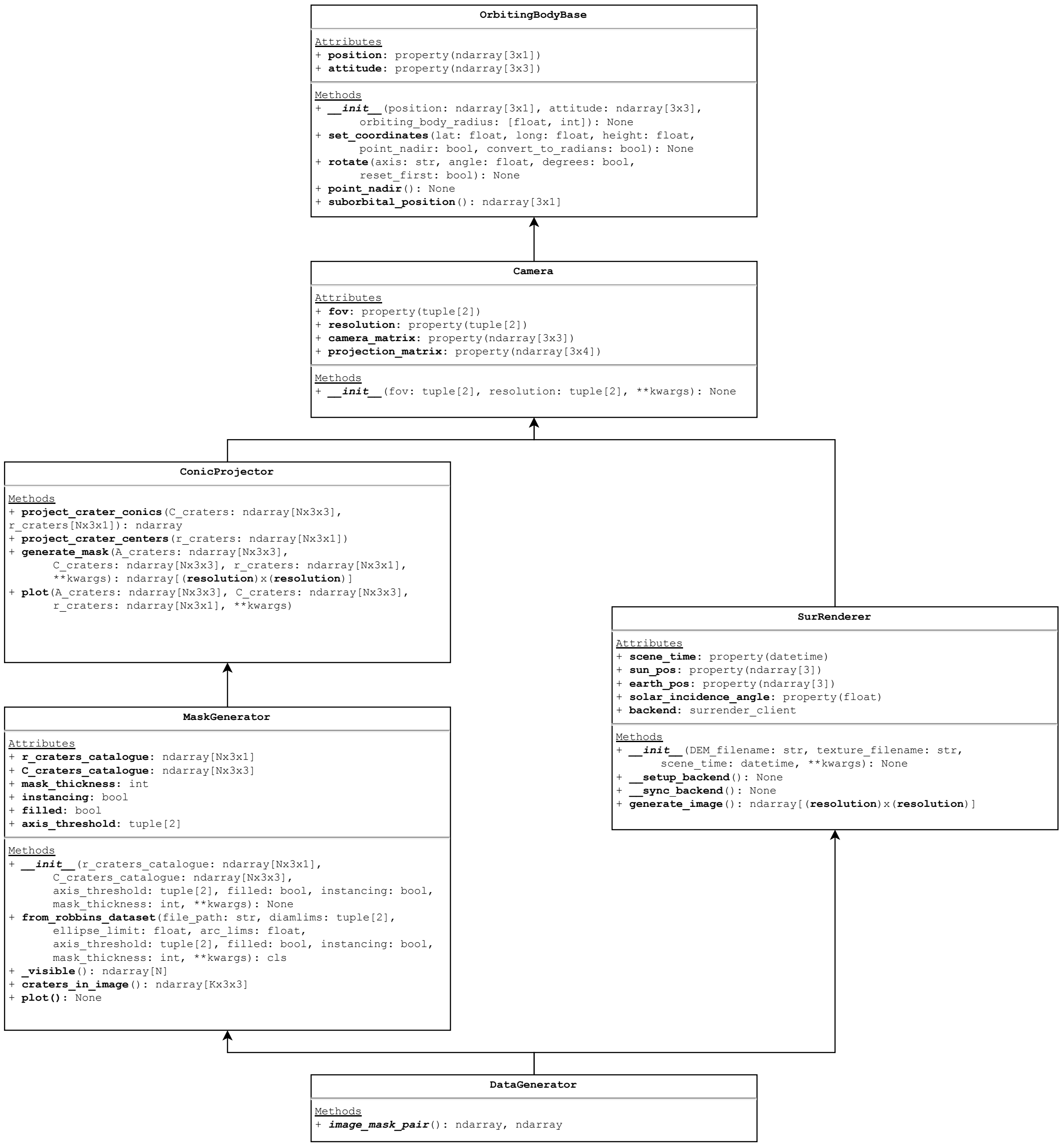

OrbitingBodyBase
Attributes
+ position: property(ndarray[3x1])
+ attitude: property(ndarray[3x3])
Methods
+ __init__(position: ndarray[3x1], attitude: ndarray[3x3], orbiting_body_radius: [float, int]): None
+ set_coordinates(lat: float, long: float, height: float, point_nadir: bool, convert_to_radians: bool): None
+ rotate(axis: str, angle: float, degrees: bool, reset_first: bool): None
+ point_nadir(): None
+ suborbital_position(): ndarray[3x1]
Camera
Attributes
+ fov: property(tuple[2])
+ resolution: property(tuple[2])
+ camera_matrix: property(ndarray[3x3])
+ projection_matrix: property(ndarray[3x4])
Methods
+ __init__(fov: tuple[2], resolution: tuple[2], **kwargs): None
ConicProjector
Methods
+ project_crater_conics(C_craters: ndarray[Nx3x3], r_craters[Nx3x1]): ndarray
+ project_crater_centers(r_craters: ndarray[Nx3x1])
+ generate_mask(A_craters: ndarray[Nx3x3], C_craters: ndarray[Nx3x3], r_craters: ndarray[Nx3x1], **kwargs): ndarray[(resolution)x(resolution)]
+ plot(A_craters: ndarray[Nx3x3], C_craters: ndarray[Nx3x3], r_craters: ndarray[Nx3x1], **kwargs)
SurRenderer
Attributes
+ scene_time: property(datetime)
+ sun_pos: property(ndarray[3])
+ earth_pos: property(ndarray[3])
+ solar_incidence_angle: property(float)
+ backend: surrender_client
Methods
+ __init__(DEM_filename: str, texture_filename: str, scene_time: datetime, **kwargs): None
+ __setup_backend(): None
+ __sync_backend(): None
+ generate_image(): ndarray[(resolution)x(resolution)]
MaskGenerator
Attributes
+ r_craters_catalogue: ndarray[Nx3x1]
+ C_craters_catalogue: ndarray[Nx3x3]
+ mask_thickness: int
+ instancing: bool
+ filled: bool
+ axis_threshold: tuple[2]
Methods
+ __init__(r_craters_catalogue: ndarray[Nx3x1], C_craters_catalogue: ndarray[Nx3x3], axis_threshold: tuple[2], filled: bool, instancing: bool, mask_thickness: int, **kwargs): None
+ from_robbins_dataset(file_path: str, diamlims: tuple[2], ellipse_limit: float, arc_lims: float, axis_threshold: tuple[2], filled: bool, instancing: bool, mask_thickness: int, **kwargs): cls
+ _visible(): ndarray[N]
+ craters_in_image(): ndarray[Kx3x3]
+ plot(): None
DataGenerator
Methods
+ image_mask_pair(): ndarray, ndarray

Figure 3.8: DataGenerator class hierarchy.

4 Crater Detection

Section 1.2 summarises the current state-of-the-art in CDAs, showing a justified interest in the application of DL-based solutions in the most recent solutions. Indeed, they hold the most promise for generalisation in challenging environments. However, the development of such a solution is only as good as the data that is used to train and test it, along with a host of other factors such as model architecture and hyperparameters. This chapter describes a novel CDA that utilises a region-based object detection network called Faster R-CNN [69] as a basis. It is modified to be able to detect the elliptical shapes of the detected instances by adding an extra Region of Interest (RoI)-head that uses the convolutional features of a classification model called ResNet [44] to predict normalised shape parameters. The extra RoI-head is a three-layer MLP which predicts normalised ellipse axis values along with their rotation. This 'ellipse regressor' was directly optimised to minimise the Gaussian Angle divergence [17] between predictions and target ellipses. Results show cutting edge precision under a wide variety of circumstances, ranging from different surface types, lighting conditions, and camera attitudes.

4.1. Model Definition

Instead of using any pixel-wise mask, a novel method of detecting craters using CNNs is introduced. Although a region-based model has been used in a crater detection system before [28], it lacks the capability to return crater shapes. Our method uses the object detection functionality and performance of a Faster R-CNN model, in combination with a custom predictor that uses the latent features of a ResNet50 backbone to generate normalised ellipse parameters. The approach is similar to that of Ellipse R-CNN [23]. Instead of having to post-process pixel-wise masks with ellipse fitting algorithms, the model can now optimise directly for ellipse shape. A more direct way of measuring model accuracy can then be introduced to train the network to generate crater shapes $\boldsymbol{A}_Q$.

4.1.1. Feature Engineering

Before the model architecture is defined, it needs to be clear what the target features for a single detected crater instance are. Starting from the image-mask pairs described in the previous chapter, a loop through all individual pixel values that indicate separate crater instances in the image is performed to generate the bounding boxes. Simply finding the extremes for the pixel-wise mask enables the generation of the bounding box descriptor B. The ellipse E that defines the crater rim has its origin at the box centre, and is further defined by its axes and rotation relative to the x-axis. See Figure 4.1 for an illustration.

To normalise the output space that the ellipse regressor predicts upon, the target features generated for E_a, E_b are log-space translations relative to the box diagonal Q (Equation 4.2 and 4.3). This creates a constrained output range which prevents the possibility of negative axes predictions, as well as to be able to infer relative axes sizes w.r.t. B rather than absolute values. The ellipse angle in radians E_θ is normalised by $\frac{\pi}{2}$ (Equation 4.4).

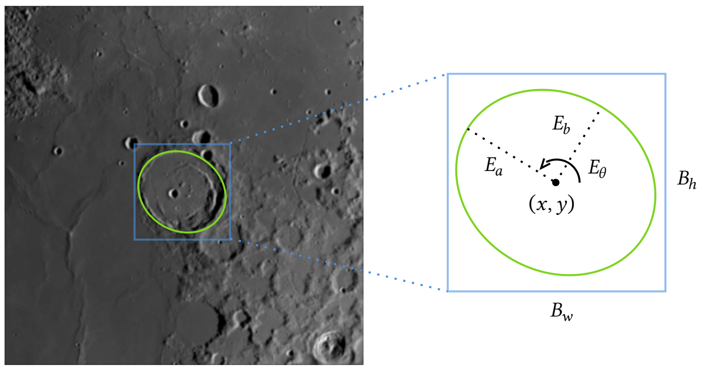


Figure 4.1: Illustration of the definition of all features related to the geometry of a single detected crater.

$$Q = \sqrt{B_w^2 + B_h^2} \tag{4.1}$$
$$\delta_a = \ln(2E_a/Q) \tag{4.2}$$
$$\delta_b = \ln(2E_b/Q) \tag{4.3}$$
$$\delta_\theta = 2E_\theta/\pi \tag{4.4}$$

To cast the target values back to pixel space, the inverse translations are as follows:

$$E_a = \frac{Q}{2}e^{\delta_a} \tag{4.5}$$
$$E_b = \frac{Q}{2}e^{\delta_b} \tag{4.6}$$
$$E_\theta = \frac{\pi}{2}\delta_\theta \tag{4.7}$$

Activation Function
The choice of activation function is important considering the desired output ranges for $\delta_a, \delta_b, \delta_\theta$. As a matter of fact, the target ellipse's semi-major or semi-minor axis cannot exceed roughly half of the bounding box' diagonal Q, and the target ellipse angle E_θ falls in the range $(-\frac{\pi}{2}, \frac{\pi}{2})$. The chosen function is therefore the hyperbolic tangent:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{4.8}$$

This squeezes the regressor output between $(-1, 1)$. Substituting this range into the possible output range means that the axis sizes can range from $(\frac{e^{-1}}{2}Q, \frac{e^1}{2}Q)$ and the angle falls between $(-\frac{\pi}{2}, \frac{\pi}{2})$. The idea is that the model has the capacity to learn the desired values from the raw output of the convolutional layers associated with the region proposals.

4.1.2. Loss Function

Unlike Ellipse R-CNN [23], the raw output for $\delta_a, \delta_b, \delta_\theta$ is not directly compared with their target values with a smoothed L_1 loss function. Instead, these outputs are cast back into the image domain using Equation 4.5 - 4.7 to form the ellipse parameters E_a, E_b, E_θ along with their centre denoted by B_x, B_y to form a conic matrix $\boldsymbol{A}_Q$ that satisfies the equation of a conic section Equation 3.15. In order to determine a goodness-of-fit for a predicted ellipse w.r.t. the target, the divergence is calculated. In other words, ellipses that are unlike in shape and position need return a higher value and vice versa. See Figure 4.2 for an illustration of the concept. Instead

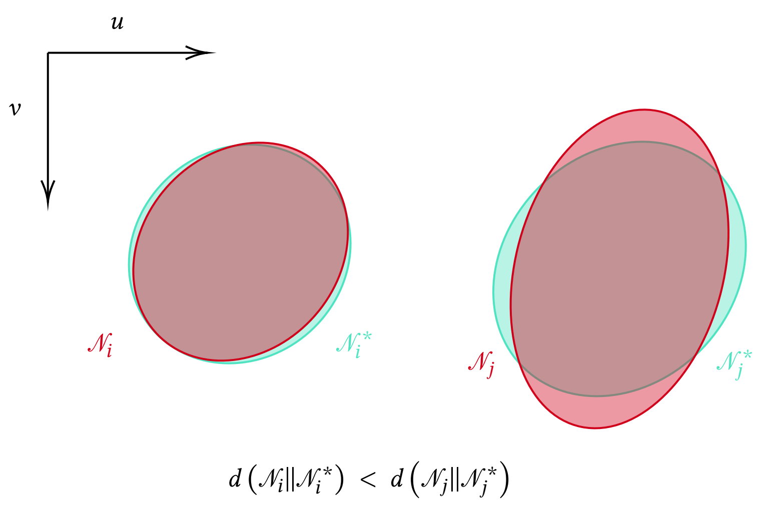

u
v
$\mathcal{N}_i$
$\mathcal{N}_i^*$
$\mathcal{N}_j$
$\mathcal{N}_j^*$
$d\left(\mathcal{N}_i \| \mathcal{N}_i^*\right) < d\left(\mathcal{N}_j \| \mathcal{N}_j^*\right)$

Figure 4.2: Illustration of prediction vs. target in fitting an ellipse to a crater rim in the image plane.

of using metrics such as Intersection-over-Union (IoU), which would require the ellipse shape to be drawn in discrete pixels, a more direct approach is desired to both speed up the loss calculation and to increase accuracy.

To achieve this, both ellipses are approached as two-dimensional Gaussian distributions $\mathcal{N}(\boldsymbol{m}, \sum)$, with $\boldsymbol{m}$ and $\sum$ denoting the mean and covariance matrix, respectively. The ellipse shape is essentially the 1σ boundary for the distribution. Approaching the geometric shape of an ellipse as a distribution allows for direct calculation of the aforementioned divergence using the Gaussian Angle distance.

To achieve this, the covariance matrices and means for both the predicted ($\boldsymbol{A}_i$) and target ellipse ($\boldsymbol{A}_i^*$) are retrieved. After scaling both to satisfy $|\boldsymbol{A}_i| = |\boldsymbol{A}_i^*| = 1$, these values are as follows:

$$\sum = \boldsymbol{A}_{33} = \begin{bmatrix} A & B/2 \\ B/2 & A \end{bmatrix} \tag{4.9}$$

$$\boldsymbol{m}_i = \begin{bmatrix} B_x \\ B_y \end{bmatrix} \tag{4.10}$$

The Gaussian Angle distance was defined to satisfy four distance metric axioms [19]. Given two ellipses in an image $\mathscr{A}_i$ and $\mathscr{A}_j$:

1. $d(\mathscr{A}_i, \mathscr{A}_j) = 0$ if $\mathscr{A}_i = \mathscr{A}_j$
2. $d(\mathscr{A}_i, \mathscr{A}_j) = d(\mathscr{A}_j, \mathscr{A}_i)$
3. $d(\mathscr{A}_i, \mathscr{A}_j) \leq d(\mathscr{A}_i, \mathscr{A}_k) + d(\mathscr{A}_j, \mathscr{A}_k)$
4. $d(\mathscr{A}_i, \mathscr{A}_j) = d(S[\mathscr{A}_i], S[\mathscr{A}_j])$

Using these properties, the distance metric is defined as follows (see [17] p. 72 - 74 for derivation):

$$d_{GA}(\mathcal{N}_i \| \mathcal{N}_i^*) = \arccos \left\{ \frac{4\sqrt{|\sum_i||\sum_i^*|}}{|\sum_i + \sum_i^*|} \exp\left[-\frac{1}{2}(\boldsymbol{m}_i - \boldsymbol{m}_i^*)^T \sum_i (\sum_i + \sum_i^*)^{-1} \sum_i^* (\boldsymbol{m}_i - \boldsymbol{m}_i^*) \right] \right\} \tag{4.11}$$

Because both the bounding box and ellipse regressors are involved in predicting $\boldsymbol{A}_i$, any changes in loss are propagated backward to further tune both RoI-heads to be capable of generating reliable ellipse predictions.

4.1.3. Architecture

The custom RoI-head capable of predicting $\delta_a, \delta_b, \delta_\theta$ was created in a similar way that Mask R-CNN creates per-pixel segmentation predictions. This meant that a 3-layer (input, hidden, output) MLP was chosen to be able to predict on the unrolled 1024 pixels, passing through a hidden layer with 512 nodes, eventually generating three outputs. The activation function between these layers is the aforementioned hyperbolic tangent. The full model architecture is shown in Figure 4.3. Not shown here are the post-processing steps f that combine prediction output B and E to form the set of output ellipse matrices $\{\boldsymbol{A}_i := f(B_i, E_i) \mid 0 \leq i < n_{det}\}$. Integrating this into the detection model is required to calculate the divergence losses per ellipse, but also adds a bit of convenience when parsing the output to the crater identification subsystem.

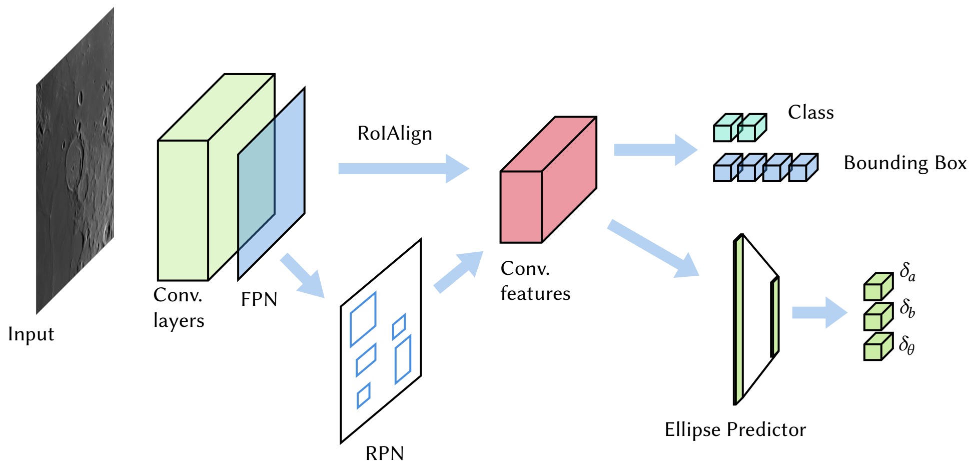


Figure 4.3: Complete model architecture.

One major benefit of integrating all post-processing steps in a single model is that the framework that was used (PyTorch) has a convenient method of exporting to shareable formats like ONNX[1], which can be used for model deployment in other frameworks such as OpenVINO[2]. This means that if a TRL 4 engineering model is to be made, it allows for rapid prototyping, since dedicated mobile AI accelerators usually come with packages that can optimise and deploy such models.

4.1.4. Training

Training the detector requires the creation of a data loader that takes the input image data along with the associated ellipses and other meta data and creates batches. This was achieved by extending the base `Dataset` class included in the PyTorch framework. Using this, a highly customisable training loop is created, where all hyperparameters given in Table 4.1 can be chosen. This finally enables the most unpredictable and challenging procedure of training any DL model: hyperparameter optimisation [90]. Essentially, it is impossible to predict exactly what the effects are when any of these parameters are changed. In this case, it was chosen to use roughly the same values used for training Faster R-CNN [69] since much of the CDA is based on it. This yields the following parameters:

Furthermore, a scheduler is used to adapt the learning rate for further fine tuning if and when the validation losses are not decreasing anymore. The training process is tracked using MLflow[3], allowing convenient separation of training sessions with associated model weights saved for later use.

[1]`https://onnx.ai`
[2]`https://docs.openvinotoolkit.org`
[3]`https://www.mlflow.org`

Name	Value
Batch size	32
Training dataset size	$80 \cdot 10^3$
Validation dataset size	$2 \cdot 10^3$
Testing dataset size	$1 \cdot 10^3$
Learning rate	0.01
Momentum	0.9
Weight decay	0.0005
Epochs	20
Optimizer	Stochastic Gradient Descent

Table 4.1: Crater detector training hyperparameters.

4.2. Results

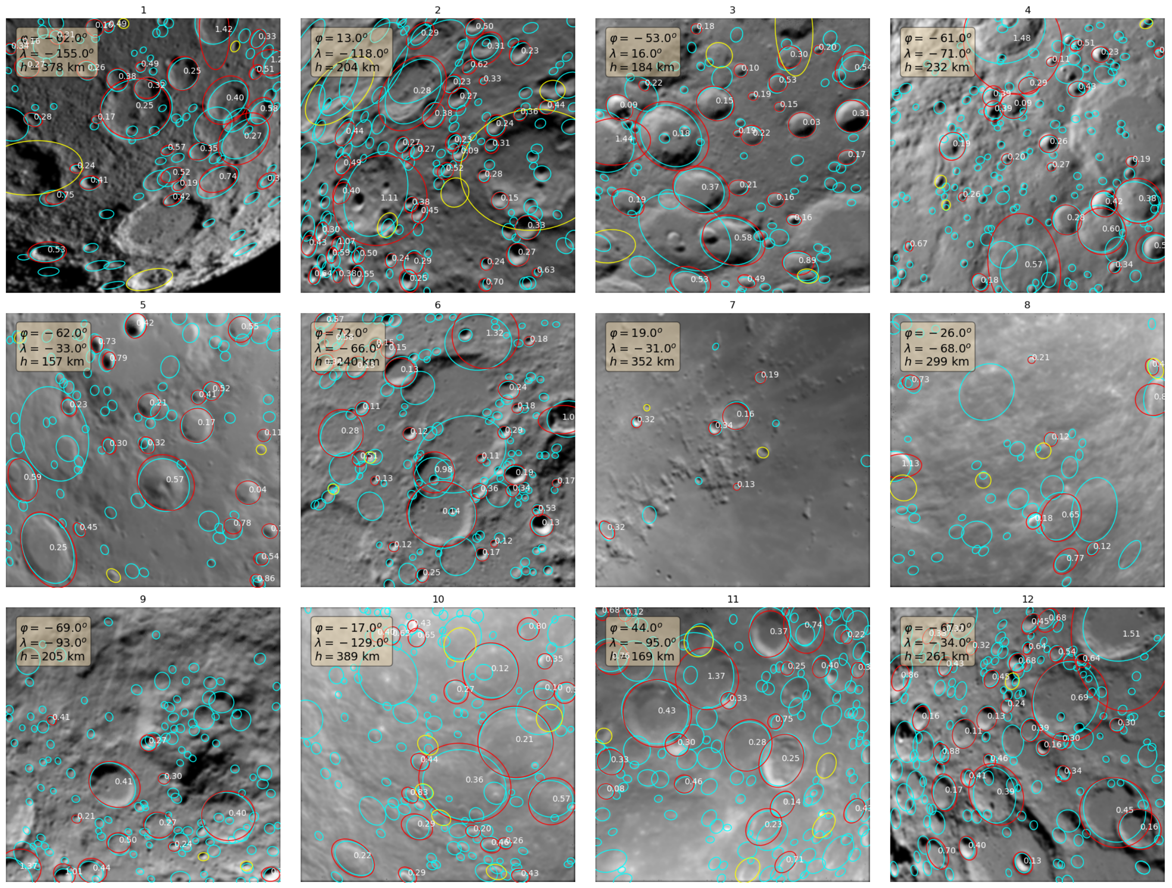


Figure 4.4: Sample of model detections (red is a True Positive, yellow is a False Positive) versus ground-truth (cyan) with associated ellipse divergence d_{GA} for every True Positive. Minimum class score is set at 75%.

To understand the output and the accuracy of the model, a sample similar to the one shown in Figure 3.7 is retrieved from the test dataset and passed through the model. Examining Figure 4.4 yields some interesting results about the behaviour of the model. Firstly, it seems that the model's predictions are mostly true positives. In other words, the CDA can separate the craters and background well. Next, the shape that is inferred from the convolutional features appears to be captured as well, with relative axis sizes and angle following the crater shape. In some cases, it appears that the model does exhibit some ellipse fitting error, and at first glance this could be caused by challenging lighting conditions (#11), out-of-shape craters (top right crater in #12), or partially visible craters (right-most crater in #6). Generally this can be attributed to a lack of discriminating features apparent in the image with which the model can predict the crater shape. It is found that predictions performed using the test dataset exhibited an average axis error of approximately 2.6 ± 1.8 [pixels]. This means that **CDT-PR-01.5** is verified.

As Figure 4.4 may already suggest, the system is functional under non-nadir-pointing attitudes. To verify this, a sample is made using the data generator with incremental pitch angles. Figure 4.5 displays the outcome, and it becomes clear that this does work as expected. The performance seems to degrade for craters that have a smaller footprint in the image due to increased ellipticity as the camera pitch angle increases. However, this can be attributed to the low resolution decreasing the amount of discriminatory features when the apparent ellipticity is very high ($\frac{a}{b} > 2$), ultimately resulting in a thin strip if pitch angle increases enough (see top of right-most image in Figure 4.5). It is hypothesised that this can be achieved given a higher resolution and more training, however this is considered outside the scope of this project. This proves the system's compliance with **CDT-PR-01.1**.

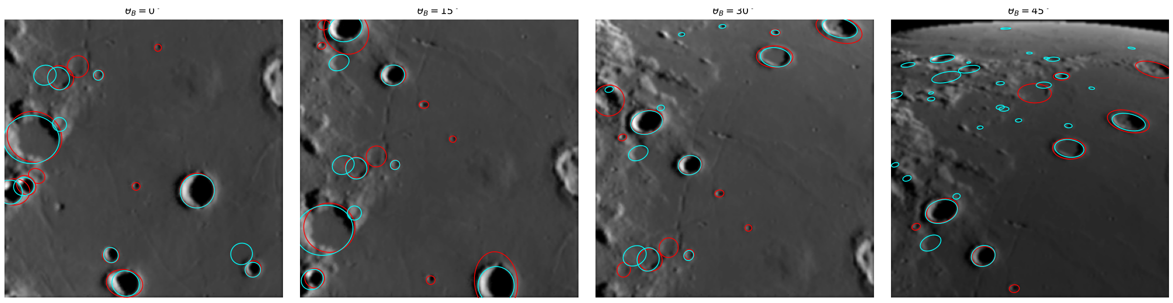


Figure 4.5: Sample of model detections in varying attitude angels, red is a detected crater, cyan is a ground-truth crater.

Figure 4.6 displays the system's capabilities under varying sun angles. Using the data generator, a series of images are made with different scene times, which result in a change in the relative position of the Sun. It is evident that the CDA has correctly learned to detect craters with different shadow profiles as they appear in the image, proving that an AI-based approach is valid. This proves the crater detection subsystem's compliance with **CDT-PR-01.2**.

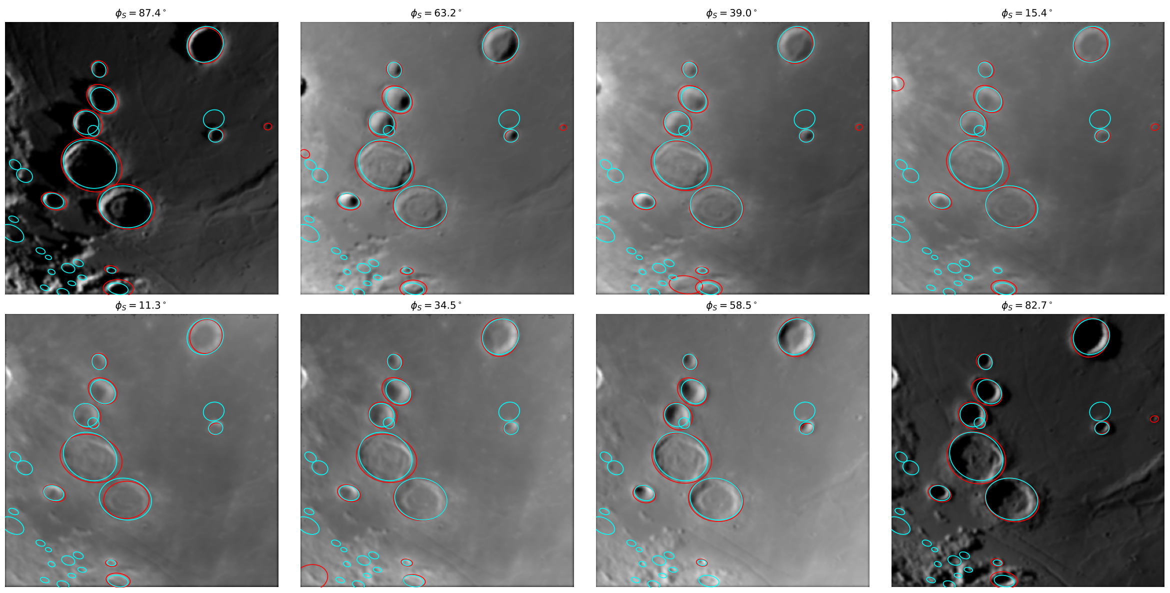


Figure 4.6: Sample of model detections in varying light conditions, red is a detected crater, cyan is a ground-truth crater. Each image is 2 days further in time, resulting in a different sun angle.

Quantifying the performance of the CDA requires a clear definition of true and false detections in the context of crater detection, as well as the value describing a goodness-of-fit. True and False Positives (TP, FP), and False Negatives (FN) are defined by the following conditional statement that is dependent on a detection's confidence value p and its IoU with its respective matched target compared to thresholds t_p and t_{IoU} :

$$\text{Detection} = \begin{cases} \text{TP}, & \text{if } p \geq t_p \wedge \text{IoU} \geq t_{\text{IoU}} \\ \text{FP}, & \text{if } p \geq t_p \wedge \text{IoU} < t_{\text{IoU}} \\ \text{FN}, & \text{if } p < t_p \wedge \text{IoU} \geq t_{\text{IoU}} \end{cases}$$

Often-used metrics are Precision P, Recall R, and F_1 [28, 51], which describe the performance of an object detector through its True and False Positives (TP, FP), and True and False Negatives (TN, FN):

$$P = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4.12}$$

$$R = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.13}$$

$$F_1 = 2\frac{P \cdot R}{P + R} \tag{4.14}$$

In object detection models, a detection is a True Positive when its bounding box has an IoU that is above a certain threshold. Besides this, the distance metric given in Equation 4.11 is also used for evaluation as it more accurately describes whether a detected crater is similar enough to its associated target. See Figure 4.7 for an illustration of the concepts.

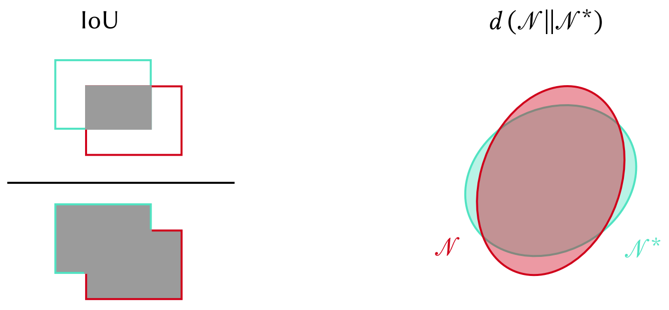


Figure 4.7: Illustration of IoU and divergence.

Precision-Recall curves can provide insight into the detector's performance [20]. The visualisation is created by calculating the Precision (Equation 4.12) and Recall (Equation 4.13) with a range of confidence thresholds. This threshold determines whether a prediction is considered part of a class. So for example if a detection has a confidence above 50%, it is accepted as a crater. For every confidence threshold, the model's Precision and Recall is calculated and plotted, with the Area Under the Curve (AUC) metric representing the Average Precision (AP).

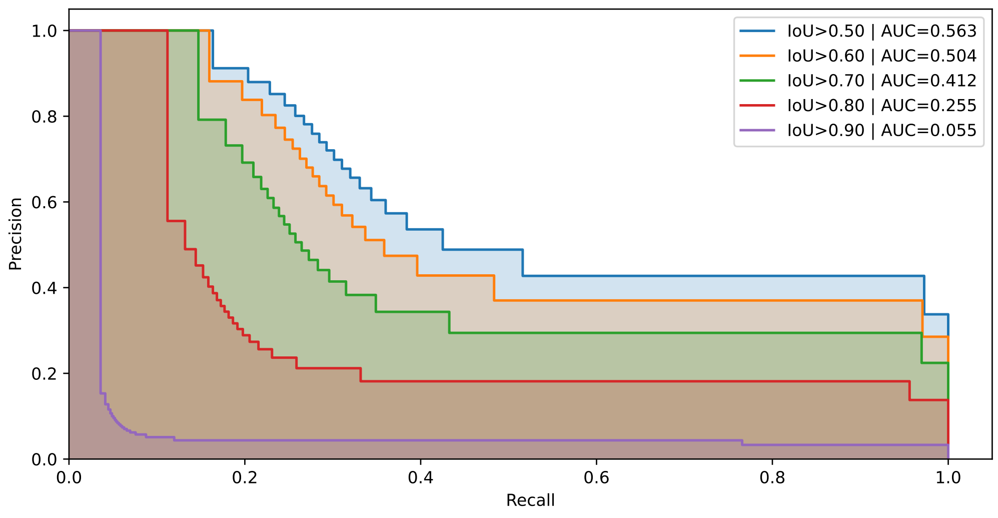


Figure 4.8: Precision-Recall curve for Ellipse R-CNN-based CDA with multiple minimum IoU required to consider a detection a true positive.

This performance is comparable to state-of-the-art object detection models like YOLOv3 [68], which is to be expected. The difference here is that the target data is not the commonly used COCO dataset [55], but rather the generated lunar scenes, which is a comparatively much easier classification task.

Besides object detection performance, the mean ellipse divergence is a key metric to give insight into how well the detector can return well-fitted ellipses describing crater rims in the image plane. The metric used is given in Equation 4.11, which returns lower values for good fits, and vice versa (see Figure 4.2). To have an idea about how these values scale, please refer to Figure 4.4. Plotting this distance metric as a function of the confidence threshold t_p returns Figure 4.9

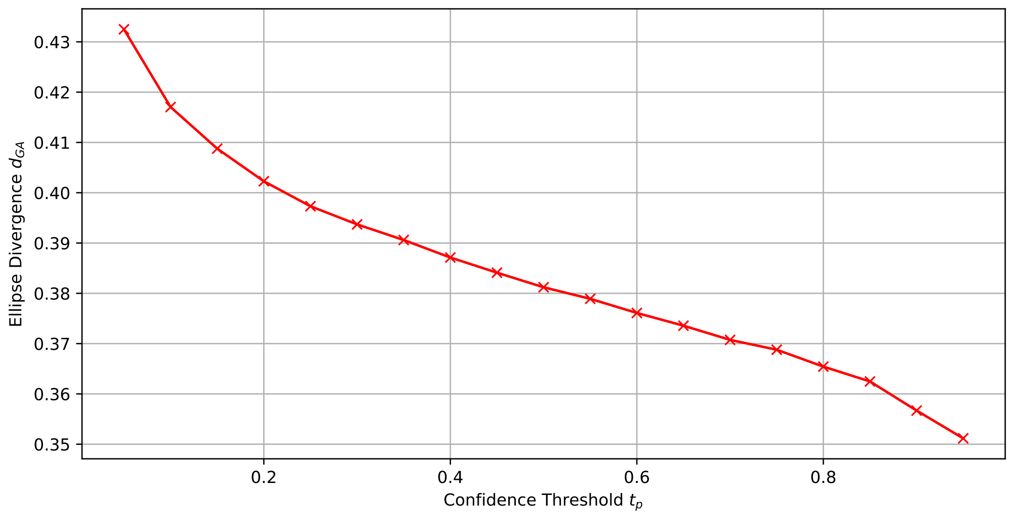


Figure 4.9: Mean ellipse divergence d_{GA} (lower is better) for all positive detections (TP + FP) plotted as a function of varying confidence threshold t_p.

The proposed CDA functions in a representative simulation environment, and has the intrinsic value of being an end-to-end solution which does not require the addition of sensitive post-processing steps. Building upon the functionality provided by the data generator described in the previous chapter, a virtually limitless amount of scenes could be created to create a large enough dataset to guarantee a generalisable model. The results present a realistic view on the difficulty of reliable and scalable crater detection, but considering the use case that this subsystem is partially fulfilling there is an acceptable amount of error as long as enough 'good enough' detections are available for the position estimation subsystem. The tests up until this point return a positive result for the performance of the model, however it does not yet prove its efficacy in a full position estimation pipeline.

5 Position Estimation

To be able to derive the spacecraft ego-position, a crater pattern identification method is implemented. The methodology is mostly based on the coplanar crater descriptor from [17] because of its capacity to work in LIS situations and under off-nadir attitudes. It fully utilises the crater shape to form discriminating features which can be matched with a database. The database is based on a k-d tree data structure that allows real-time nearest neighbour searching for multidimensional trees [10]. Matched craters form a system of equations that is solved using RANSAC, after which a verification step is applied to remove reprojected outliers. Results show a match rate of roughly 14% of all cases, albeit with accuracy that generally falls below 4 [km] when using significantly perturbed pseudo-measurements.

5.1. Coplanar Invariants for Ellipse Triads

Creating discriminating features from a collection of conics is a topic that has been discussed in the CV domain for some time [31, 38, 47, 66], but has only recently been applied to crater pattern matching [17, 58]. The essence of the method is that there exists a unique pair of numbers for a pair of coplanar ellipses (a conic) that are invariant to affine transformations. This is a very useful property that can be exploited to create the desired discriminating features which is independent of camera pose. To further increase the uniqueness of the descriptor, a triad of craters is selected to calculate seven absolute invariants which jointly make up the discriminating feature. In the following subsection a summarised derivation is given.

5.1.1. Theory

The joint invariants of two conics $\boldsymbol{A}_i, \boldsymbol{A}_j$ that satisfy Equation 3.15 and $|\boldsymbol{A}_i| = |\boldsymbol{A}_j| = 1$ are given by [17]:

$$I_{ij} = \text{Tr}[\boldsymbol{A}_i^{-1}\boldsymbol{A}_j] \tag{5.1}$$

$$I_{ji} = \text{Tr}[\boldsymbol{A}_j^{-1}\boldsymbol{A}_i], \tag{5.2}$$

with Tr denoting the trace of a matrix. Proving that this is invariant to affine transformations can be done by applying the same transformation T to both conics [43] as $\boldsymbol{A}_i' = T^T\boldsymbol{A}_iT$ and $\boldsymbol{A}_j' = T^T\boldsymbol{A}_jT$, yielding the following equality for the joint coplanar invariants:

$$I_{ij}' = \text{Tr}[T^{-1}\boldsymbol{A}_i^{-1}T^{-T}T^T\boldsymbol{A}_jT] = \text{Tr}[\boldsymbol{A}_i^{-1}\boldsymbol{A}_j] = I_{ij}, \tag{5.3}$$

with the same holding for the reverse direction I_{ji}'. This means that theoretically, two different cameras observing a pair of coplanar craters can derive the same invariant. Considering that craters on a sphere are not coplanar, it is only possible to derive them for *approximately* coplanar craters. Extending the same property to a triad of ellipses $\boldsymbol{A}_i, \boldsymbol{A}_j, \boldsymbol{A}_k$ yields $2 * 3 + 1 = 7$ invariants, where the seventh value is given by [17]:

$$I_{ijk} = Tr[(\boldsymbol{A}_j + \boldsymbol{A}_k)^* - (\boldsymbol{A}_j - \boldsymbol{A}_k)^*]\boldsymbol{A}_i. \tag{5.4}$$

See Figure 5.1 for an illustration of the concept behind the coplanar invariants of three conics.

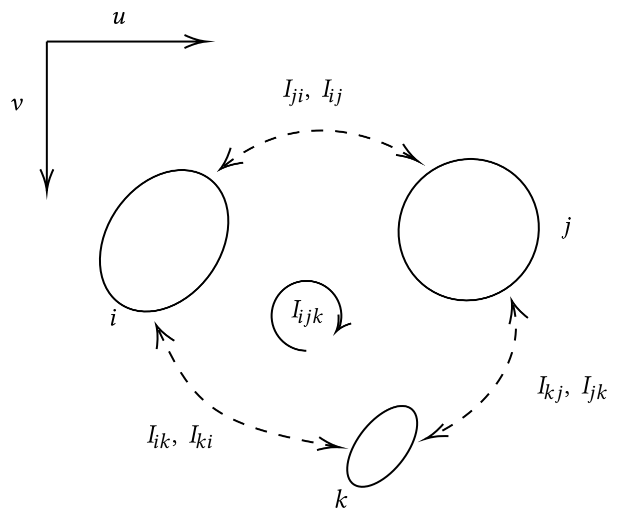


Figure 5.1: Illustration of the definition of coplanar invariants for a triad of ellipses.

Combined these form a robust descriptor that can be used in combination with the proposed CDA since it directly outputs the detected crater ellipses in the image plane. See subsection A.3.1 for the entire derivation based on the implementation given in [17]. Given a set of craters in the image plane, an array of matchable features in the shape of seven-element descriptors is created by iterating through them using an Enhanced Pattern Shifting sequence [8]. This method was first developed for star trackers that similarly needed groups of three stars to feed into the identification subsystem, and was optimised to create as many unique triads in a list.

5.1.2. Verification

To verify the fact that the coplanar invariants function in practice, a series of tests using the data generator was performed. First, an example of the descriptor is made for a single triad at varying altitudes, seen in Figure 5.2.

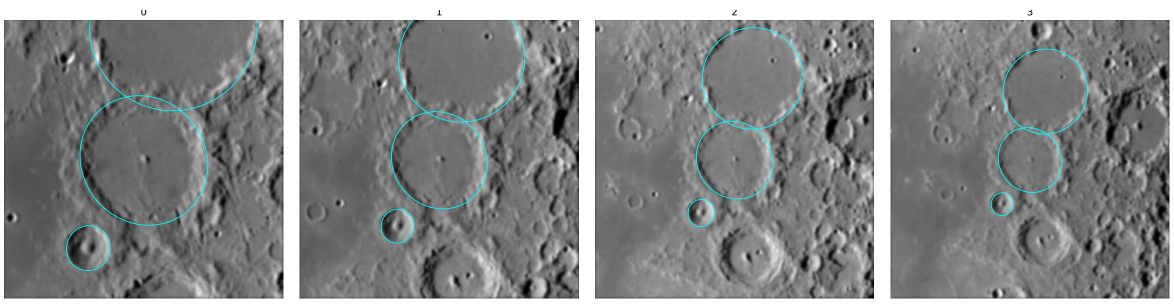


Figure 5.2: Crater triad at four different altitudes: 300, 400, 500, and 600 km.

Altitude Difference	ΔI_{ij}	ΔI_{jk}	ΔI_{ki}	ΔI_{ji}	ΔI_{kj}	ΔI_{ik}	ΔI_{ijk}
100km	-0.9%	-0.5%	-1.5%	-0.22%	-0.49%	-0.035%	-2.2%
200km	-1.4%	-0.8%	-2.5%	-0.35%	-0.79%	-0.059%	-3.7%
300km	-1.8%	-1.0%	-3.1%	-0.44%	-0.99%	-0.077%	-4.6%

Table 5.1: Seven-element descriptor value change sensitivity to altitude changes w.r.t. the descriptor for the first image in Figure 5.2.

Performing the same test with only attitude changes yields a completely identical seven-element descriptor for all cases. However, camera position and attitude is not the main factor that impacts accuracy, as this is most likely the accuracy with which ellipses are fit to a crater rim. To test this hypothesis, a Monte Carlo simulation was prepared that creates random scenes above the Moon similar to how the data generator does (see Table 3.1). Using Enhanced Pattern Shifting to generate triads of craters in the image plane, for every trial a subset of the crater was selected to form the seven-element descriptor for. After this, the artificial noise drawn from Table 5.2 was added to a copy of the exact same subset to allow for a direct comparison of values. The result of 5000 tests is displayed in Figure 5.3 and splits the error for the clockwise descriptors, counter-clockwise descriptors, and the triad descriptor I_{ijk}.

Name	Symbol	Unit	Distribution
Axis	a, b	pixels	$\mathcal{N}(0,1)$
Angle	θ	°	$\mathcal{U}(-10,10)$
Centre	$\boldsymbol{m}$	pixels	$\mathcal{N}(0,1)$

Table 5.2: Distributions for seven-element descriptor sensitivity testing.

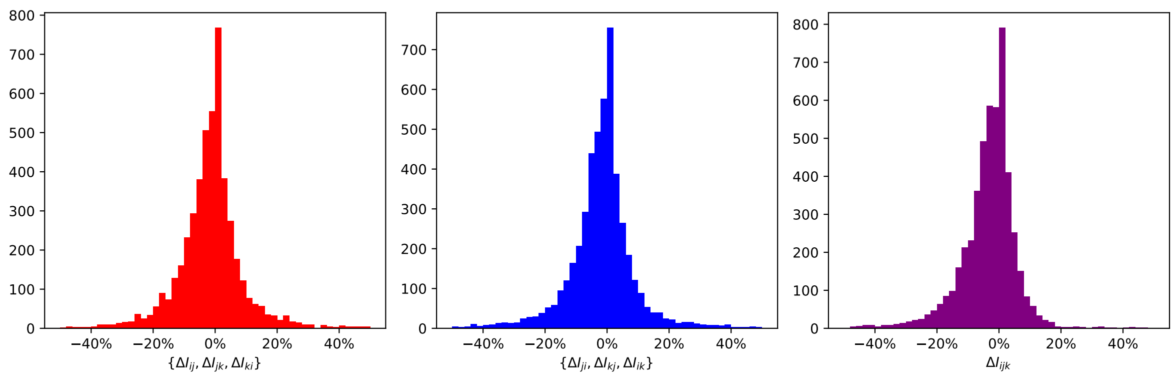


Figure 5.3: Seven-element generation sensitivity to ellipse fitting in 5000 random camera scenes with at most 20 craters present. This is the change in value after the artificial error is introduced.

It is apparent that the coplanar invariants are relatively sensitive to ellipse fitting error. However due to the seven dimensions of the descriptor it was hypothesised that the probability of identifying a detected triad is high enough to be used in a navigation solution. To prove this, an end-to-end database with querying functionality was developed.

5.2. Crater Database

Building a crater database is achieved by creating a searchable index that can reliably match a generated 'key' to a triad of craters. This key index consists of seven-element descriptors that are built ahead of time, and as illustrated by the previous section, the same descriptors can subsequently be made in real-time for a set of detected craters. Once a set of potential matches has been found

5.2.1. Generation

Creating a readily searchable database for a crater pattern matching algorithm was achieved by taking the set of craters from a list of known craters (e.g. from [71]) and finding crater triads that are within a certain range that would allow the coplanar assumption to hold. In our implementation this was achieved by using functions from [64], where the Cartesian coordinates of the crater set and inserting them into a three-dimensional k-d tree to query each crater for their neighbours within a radius of 100km, to eventually return a sparse connection matrix containing all 'connections' between crater pairs. This sparse matrix forms the basis of a graph structure which was then used to find all cliques [59] with a size of three, which corresponds to all crater triads. Depending on how large the crater set is and how dense the connection matrix is this may take a long time, so realistically a database will not be created at the scale of the entire Moon, but rather a sub-region. This deviates slightly from the definition of a LIS-situation, however it is reasonable to assume that the state estimation of a spacecraft has not been degraded to such a degree that it is unaware which side of the Moon it is above.

Then, by iterating through all crater triads we can find the seven-element descriptor by placing a 'virtual' camera above the centroid of the triangle formed by the three crater centres. By then projecting all three craters into the image plane they are intrinsically coplanar as they appear in the image plane, which is two dimensional. See Figure 5.4 for an illustration of the process.

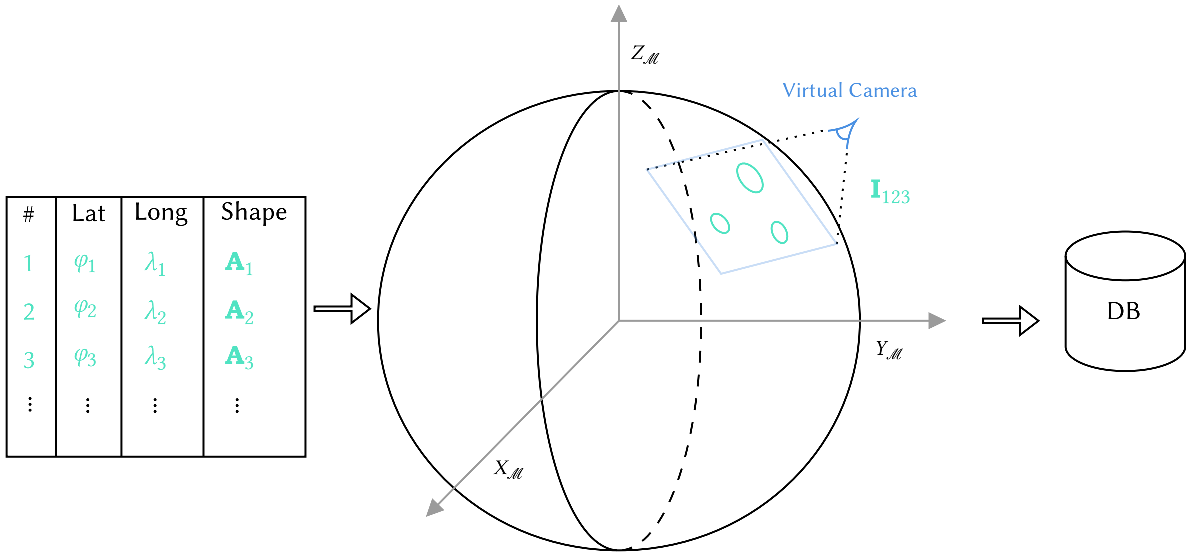


Figure 5.4: Illustration of the database generation process, where neighbouring craters form a triad are 'captured' by a virtual camera, from which the seven-element descriptor is derived and stored in the database.

The six pairwise invariants are sorted in a clockwise manner, and for each triad the crater i in $\{i, j, k\}$ is determined by ensuring that I_{ij} is the lowest value of $\{I_{ij}, I_{jk}, I_{ki}\}$. After finding and sorting all the invariant descriptors $\boldsymbol{I}$, a seven-dimensional k-d tree is created to form the final indexing data structure that is used to be able to identify craters as they appear in a detection with the catalogue.

5.2.2. Matching Functionality

Matching is achieved by querying the k-d tree that forms the basis of the database. The core functionality is provided by the `KDTree` class from [85], and allows for direct nearest-neighbour(s) searching with a configurable number of returned values along with the Cartesian distance between the detection triad's key and the database entry's key. One drawback of this implementation is that it does not allow for a per-axis (of the seven-element descriptor) normalised distance, achievable by using a weighted Minkowski distance metric. Such functionality would most likely have to be implemented with a static language such as C++ for performance reasons, which is not an unrealistic assumption in the likely event that this system is ported to an embedded target.

5.2.3. Ego-position from Corresponding Coplanar Conics

Given a set of matched craters in the image plane

$$M = \{\boldsymbol{A}_i \mid 0 \leq i < n_{det}\},$$

and a set of triad indices using Enhanced Pattern Shifting

$$K = \{[i, j, k] \mid i, j, k < |M|, i \neq j \neq k\},$$

a matchable pattern is created. This is constrained by a hard limit, since the amount of triads rises exponentially with the amount of detected craters. This is the *key* that consists of a set of seven-element descriptors per triad - the aforementioned corresponding features. Using the properties of the k-d tree that forms the database, a set of N_{query} nearest neighbours per triad are given as

$$Q_i = \{[\boldsymbol{A}_i^*, \boldsymbol{A}_j^*, \boldsymbol{A}_k^*]_n \mid 0 \leq n < N\}_i,$$

and are passed to the ego-position and match verification subsystem, which reorders the set of craters from the query and filters based on distance and removes duplicate entries. Using the methodology from [17], a linear system of equations is formed using the known camera matrix, spacecraft attitude, and a filtered subset of the detected craters. The query also returns the associated matched selenographic positions $\boldsymbol{x}^{(c)}_{\mathscr{M}_i}$, which means that the spacecraft position can be derived by essentially finding a least-squares solution to Equation 5.6 for $\boldsymbol{x}_B$. An abridged version of the setup of this system of equations is as follows:

$$\boldsymbol{B}_i = \boldsymbol{T}_B \boldsymbol{K}^T \boldsymbol{A}_i \boldsymbol{K} \boldsymbol{T}_B^{-1}, \tag{5.5}$$

$$\begin{bmatrix} \boldsymbol{S}\boldsymbol{T}^M_{E_0}\boldsymbol{B}_0 \\ \vdots \\ \boldsymbol{S}\boldsymbol{T}^M_{E_n}\boldsymbol{B}_n \end{bmatrix} \boldsymbol{x}_B = \begin{bmatrix} \boldsymbol{S}\boldsymbol{T}^M_{E_0}\boldsymbol{B}_0\boldsymbol{x}^{(c)}_{\mathscr{M}_0} \\ \vdots \\ \boldsymbol{S}\boldsymbol{T}^M_{E_n}\boldsymbol{B}_n\boldsymbol{x}^{(c)}_{\mathscr{M}_n} \end{bmatrix}, \tag{5.6}$$

using S from Equation 3.18. This system of equations is then solved for the spacecraft position $\boldsymbol{x}_B$. The first iteration of the state estimation is then used with the known spacecraft attitude $\boldsymbol{T}_B$ to create projection matrix P (Equation 3.8). After that is performed, a second iteration of Equation 5.5 and 5.6 is performed with a filtered subset of the detected craters based on the Gaussian Angle divergence (Equation 4.11) calculated between a reprojected version of the matched crater and its associated detected crater. This reprojection filter is applied as follows [17]:

$$\text{Hypothesis}(\mathscr{A}_i = \tilde{\mathscr{A}_i}) = \begin{cases} \text{Accept}, & \text{if } d^2_{GA_i}/\sigma^2 \leq 13.276 \\ \text{Reject}, & \text{otherwise} \end{cases}, \tag{5.7}$$

with

$$\sigma \approx \frac{0.85}{\sqrt{a_i b_i}} \sigma_{pix}. \tag{5.8}$$

σ_{pix} is a tuneable parameter that sets the strictness of the reprojection match verification. The full ego-position pipeline is shown in Figure 5.5.

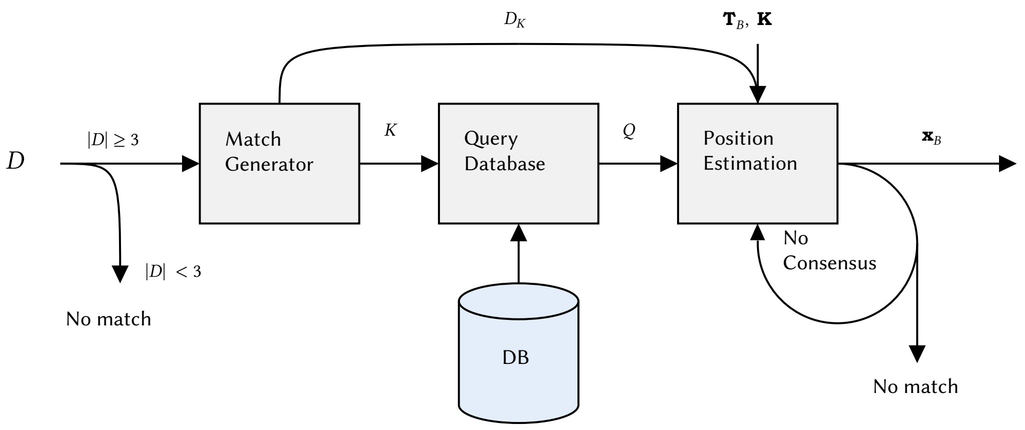


Figure 5.5: Illustration of the match verification and ego-position estimation pipeline.

During development it became apparent that using just a least-squares solution did not yield the desired accuracy that was given in [17]. It was hypothesised that this was caused by the relatively high fraction of outliers that are still present after the database query and subsequent filtering steps.

Random Sample Consensus

To solve Equation 5.6 under the presence of outliers, a method called Random Sample Consensus (RANSAC) was employed. This is a well-known algorithm used for fitting linear least-squares models with a large amount of outliers [30] and is thus potentially a solution for reliably solving Equation 5.6. This increases the system's performance in the presence of erroneous matches, and therefore increases robustness. This is a novel approach, and builds upon the work presented in [17]. RANSAC functions by fitting a linear model to a random subset of a given system of equations, it then assesses whether enough inliers are present to consider the model fitted. An illustration of the performance difference between RANSAC and linear regression is shown in a two-dimensional case in Figure 5.6.

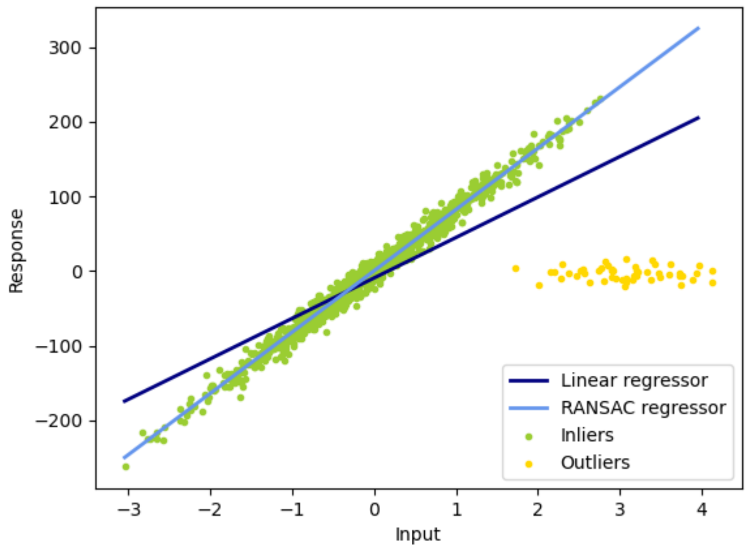


Figure 5.6: Illustration of the difference between RANSAC and linear regression on a two-dimensional system of equations with significant amounts of outliers, reprinted from [4].

5.3. Results

The complete code implementation can be found on the online repository of the project[1]. A series of Monte Carlo simulations were performed to assess the robustness of the position estimation algorithm. These tests are performed for a LIS scenario, and span over an equilateral region above the Moon from 0° to 30° for both lat- and longitude. The configuration parameters are as follows:

Name	Symbol	Unit	Distribution
Axis perturbation	$a_{\text{det}}, b_{\text{det}}$	pixels	$\mathcal{N}(0, \sqrt{2})$
Angle perturbation	θ_{det}	°	$\mathcal{N}(0, 20)$
Centre perturbation	$\boldsymbol{m}_{\text{det}}$	pixels	$\mathcal{N}(0, \sqrt{2})$
Spacecraft latitude	φ_B	°	$\mathcal{U}(0, 30)$
Spacecraft longitude	λ_B	°	$\mathcal{U}(0, 30)$
Spacecraft altitude	h_B	km	$\mathcal{U}(50, 200)$
Spacecraft roll	ϕ_B	°	$\mathcal{U}(-180, 180)$
Spacecraft pitch	θ_B	°	$\mathcal{U}(-10, 10)$
Spacecraft yaw	ψ_B	°	$\mathcal{U}(-10, 10)$

Table 5.3: Distributions for perturbed ellipses in position estimation Monte Carlo trials.

Given the stochastic nature of the input data, the output of the position estimation pipeline is expected to yield position error estimations that is partially unpredictable. The proposed system is built to work in tandem with an estimator that accept measurements with statistical noise (like an EKF), hence the need for a thorough analysis of the error statistics. Starting with the total Cartesian position error, it becomes clear that the estimator's performance generally falls within the sub-5km range. Considering the large volume of space that is used to randomly initialise the spacecraft position (roughly 1000x1000x150km), this is an impressive result. See Figure 5.7 for the error distribution.

Decomposing the error into vertical and horizontal components yields Figure 5.8 and Figure 5.9. Similar to the total accuracy $||\Delta\mathbf{x}||_2$, all components exhibit an unbiased error characteristic. This is favourable considering state estimation filters usually work best with zero-mean measurement updates [88].

Similar to the radial error statistics given in Figure 5.8, there seems to be no clear indication that the position estimation algorithm is biased in the latitude- and longitude-direction in Cartesian space. This means that

[1]`https://github.com/wdoppenberg/crater-detection/tree/main/src/matching`

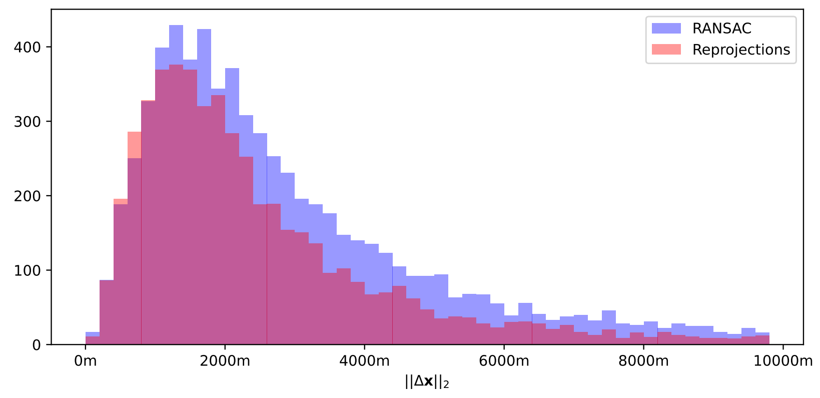


Figure 5.7: Total Cartesian position error ($||\Delta \mathbf{x}||_2 = ||\mathbf{x}_{\text{true}} - \mathbf{x}_{\text{pred}}||_2$) statistics for 11654 RANSAC inliers (blue), and 6617 verified matches through reprojection (red) with $\sigma_{pix} = 5$.

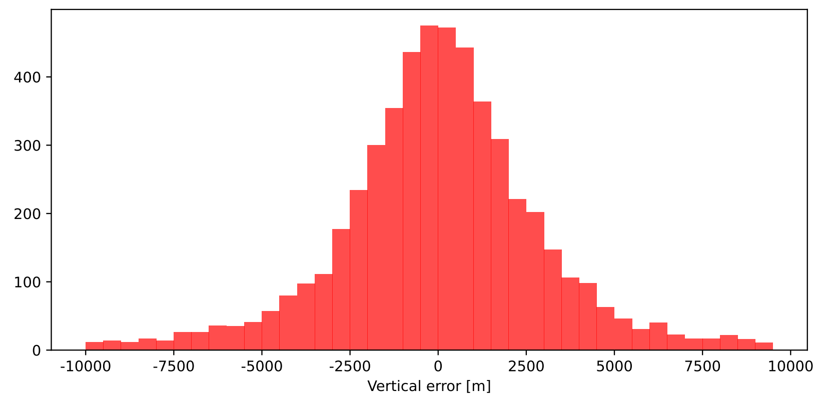


Figure 5.8: Total radial position error statistics for 6617 verified matches through reprojection with $\sigma_{pix} = 5$.

if the end-to-end system expresses such bias, it would have to be caused by the CDA subsystem. The error distribution matches the error introduced in the pseudo-measurements. Following from Figure 5.9, horizontal error statistics do not seem to skew in either longitude or latitude direction, and has a lower magnitude than radial error. It is hypothesised that this is caused by the relatively higher impact of ellipse fitting error on vertical position estimation compared to horizontal position estimation. This is caused by the perceived visual angle being relatively unaffected by a change in altitude if high enough above the Lunar surface.

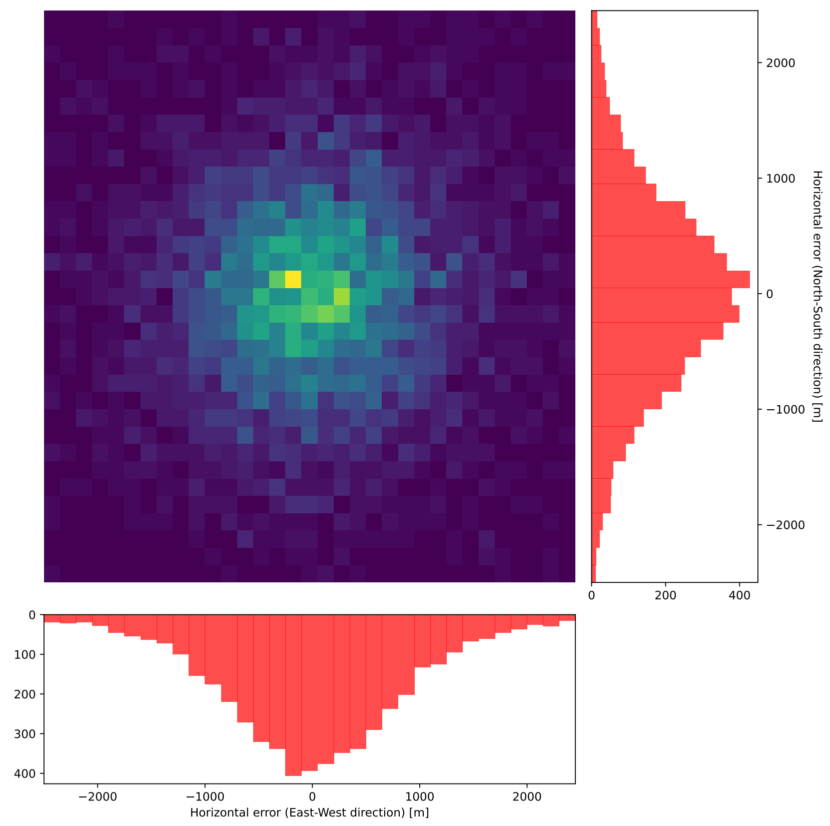


Figure 5.9: Horizontal error statistics along latitude and longitude directions.

To examine the effect of the amount of matched craters used to regress the system of equations given in Equation 5.6, a box plot showing the system's accuracy as a function of the amount of verified craters is made in Figure 5.10. No significant relation between accuracy and verified craters can be seen in this metric, however it is a testament to the methodology introduced by [17] working with relatively low amounts of matches.

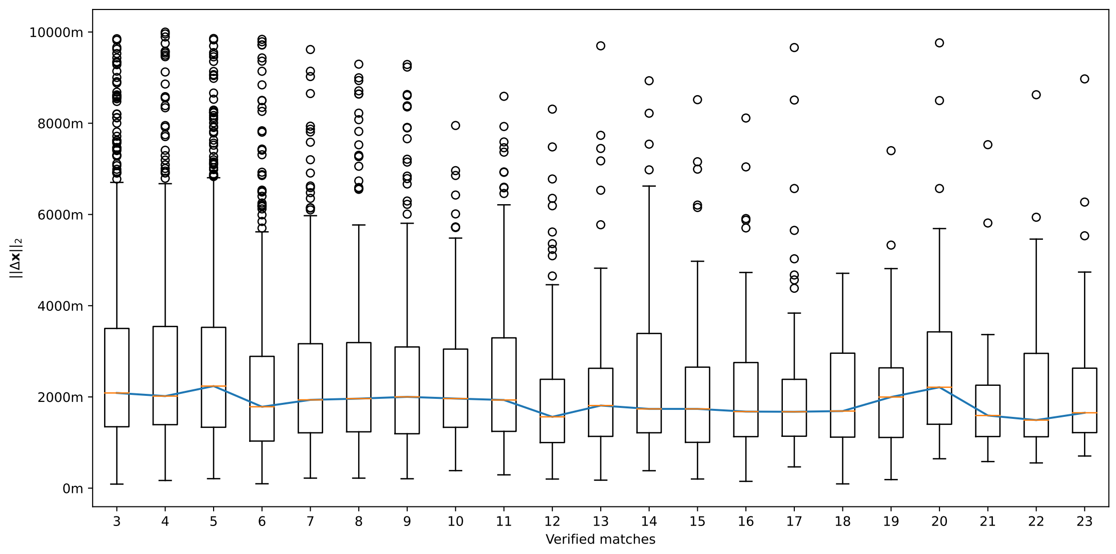


Figure 5.10: Position estimator accuracy vs. number of verified crater matches in image.

As mentioned before, altitude is a factor in the system's performance. This is further demonstrated by Figure 5.11, indicating a relation between altitude ranges and the system's accuracy.

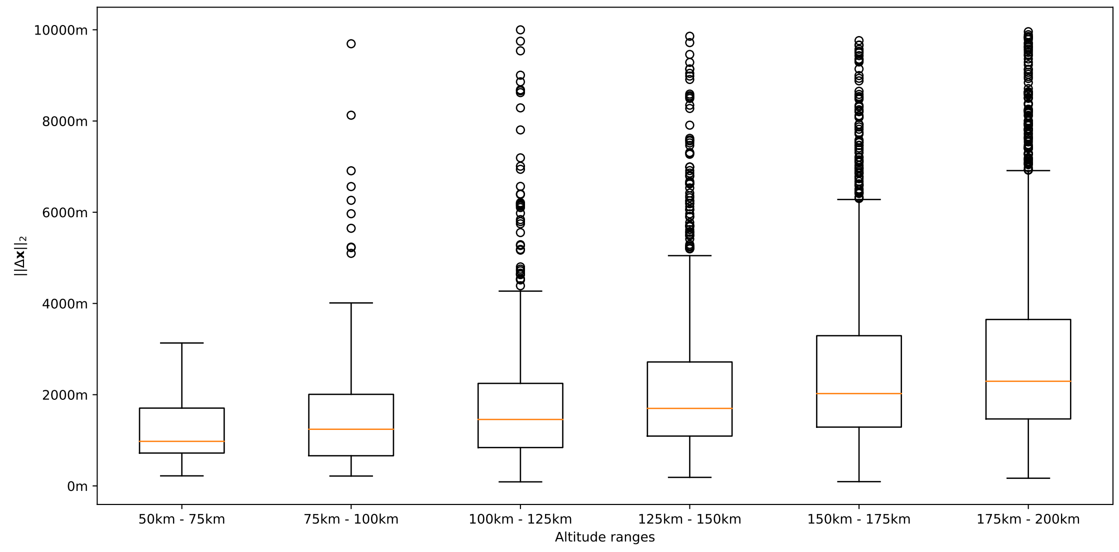


Figure 5.11: Position estimator accuracy vs. altitude ranges.

The crater identification and position estimation subsystem has been tested with highly perturbed measurements, and its performance has been characterised. It has been shown that the match rate of the identification subsystem is significantly lower than the results described in [17], which could point to an underperforming implementation or a more challenging test case. The former is possibly caused by the ad hoc methodology for querying the database, which, as mentioned before, requires further development and verification in future work. The latter could be caused by this testing method being more thorough and therefore less forgiving for the proposed subsystem. Both cases need to be investigated in later development stages.

Combined with the results of the novel CDA described in chapter 4 it is hypothesised that combining these subsystems will result in a working TRN solution. The next chapter describes the tests and results of the entire system to validate the set requirements for both components.

6 End-to-End System Performance

Assessing the proposed TRN system fully requires the CDA and position estimation subsystems to be combined into a single pipeline that can essentially accept a (simulated) image of the Lunar surface and return an absolute position estimation. To verify the capabilities as described in the previous chapters, its general performance will be assessed on random positions above a sizeable region above the Moon, which is similar to the procedure used to assess the position estimation subsystem. Instead of using pseudo-measurements, the Ellipse R-CNN-based CDA will now be used to derive crater rim shapes from simulated camera input.

6.1. Instantaneous Position Estimation Performance

The main Key Performance Indicator (KPI) for the complete system is the total Cartesian position error $||\Delta \boldsymbol{x}||_2$. This is reflected in Table 2.1, listing the maximum error for a full state estimation subsystem. The performed tests are a combination of the methods used in chapter 4 and 5. This should give insight into the effects of exogenous variables related to the scene as well as internal tuneable system settings. Just like the position estimation performance test, trials are generated by drawing from probability distributions. Using the data generator, a completely unique simulated camera output is generated for a single forward pass. This ensures that the CDA is not predicting on data it has seen before. Again, a region above the moon was chosen so as to limit the time required to build the database, but the region is large enough to still consider it a LIS situation. The minimum altitude is set at 100 km because below that value the resolution of the DEM used to generate the artificial scenes becomes too low to accurately reflect reality. It is hypothesised that the system should work with similar performance characteristics at lower altitudes as it does in this test. Future tests with higher resolution data would have to confirm this.

Name	Symbol	Unit	Distribution
Spacecraft latitude	φ_B	°	$\mathcal{U}(0, 30)$
Spacecraft longitude	λ_B	°	$\mathcal{U}(0, 30)$
Spacecraft altitude	h_B	km	$\mathcal{U}(100, 200)$
Spacecraft roll	ϕ_B	°	$\mathcal{U}(-180, 180)$
Spacecraft pitch	θ_B	°	$\mathcal{U}(-10, 10)$
Spacecraft yaw	ψ_B	°	$\mathcal{U}(-10, 10)$
Minimum ground truth craters	n_{GT}	–	10
Confidence threshold	t_p	–	75%
Maximum reprojection deviation	σ_{pix}	–	4
Catalogue diameter range	D_{cat}	km	$[4, 40]$

Table 6.1: Parameters for end-to-end system position estimation accuracy verification through Monte Carlo trials.

Using the parameters from Table 6.1, 50000 trials were performed in roughly 2.5 hours to give a complete overview of the system's performance.

6.1.1. Results

The tests yielded 8089 verified position estimations that can be considered a verified match, resulting in a match rate of roughly 16%. This is not necessarily high, however it remains to be seen whether this translates to low performance in a real-time test with multiple measurements per second. The accuracy with which the system is able to infer ego-position estimations exceeds the values retrieved from the stress tests in chapter 5. The median error for the total set of verified position estimations is ≈ 800 [m]. It needs to be emphasised that this is from a LIS situation, meaning no prior information other than the camera input and spacecraft attitude is known.

Figure 6.1 displays the total system position error, and it is clear that the system has a better accuracy than what was yielded during the pseudo-measurement tests. This means that the crater detection subsystem has managed to be accurate enough for use in a crater pattern matching technique using the elliptical shape of craters. To examine the source of the error further, the same plots as in chapter 5's results section are generated.

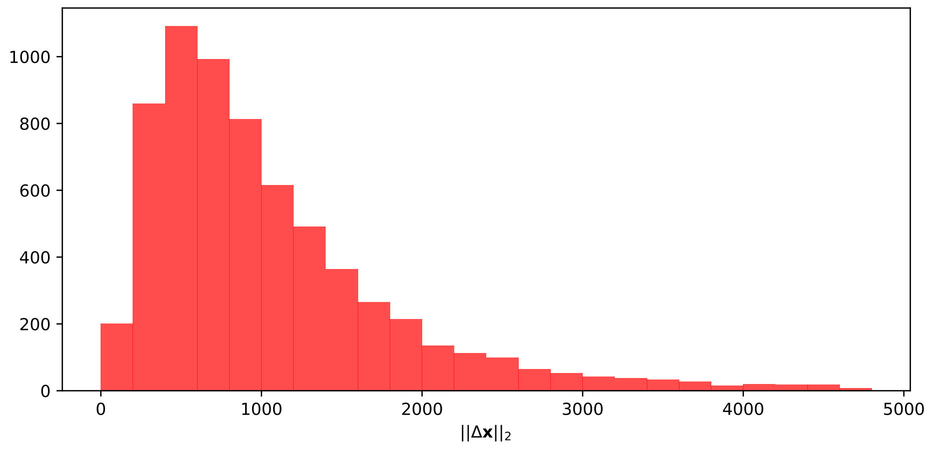


Figure 6.1: Total Cartesian position error for 8089 position estimations made by the end-to-end TRN system.

The total radial error statistics visible in Figure 6.2 display a very slight bias above 0, but the amount is not enough to draw conclusions with. This result further confirms the suspicion of the model performing above the expected results from the pseudo-measurement tests.

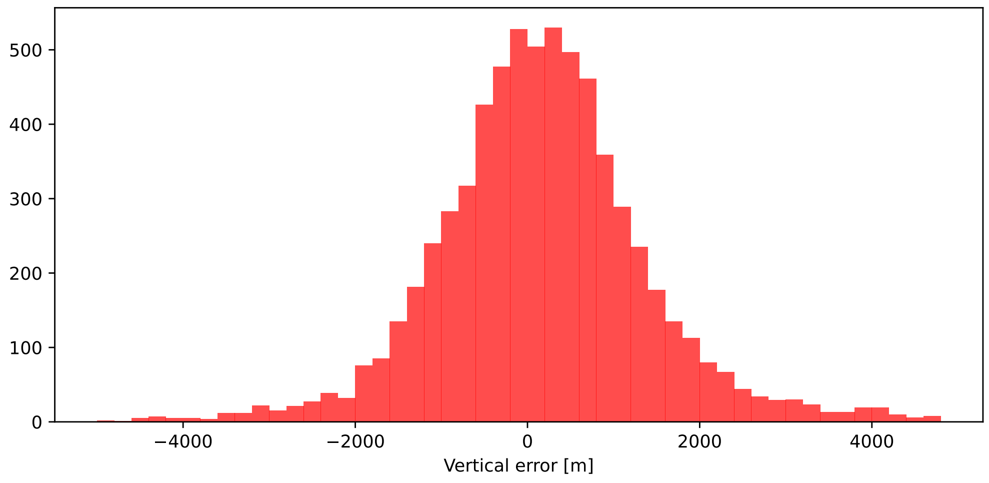


Figure 6.2: Total radial position error for 8089 position estimations made by the end-to-end TRN system.

To further analyse the system performance for verifying **CID-PR-02.1**, the horizontal error is given in a two-dimensional histogram in Figure 6.3. Again, the error appears to exhibit no bias, and appears to be of a lower magnitude than the radial position error.

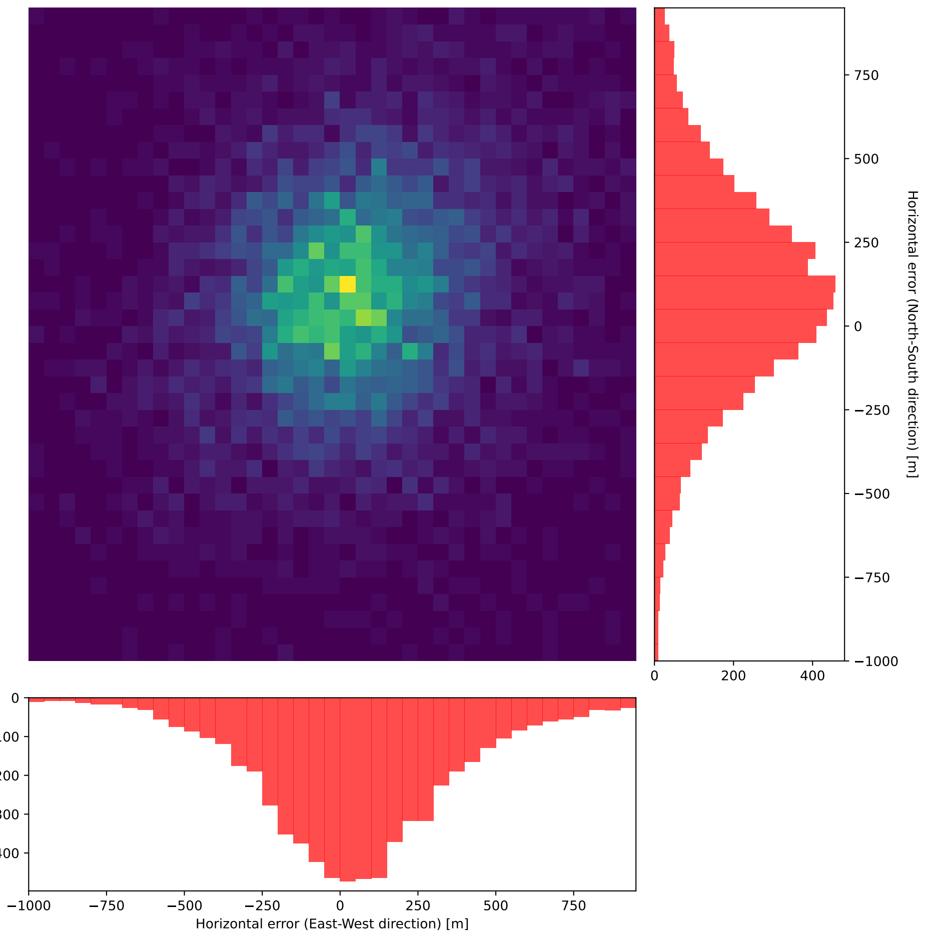


Figure 6.3: Total horizontal error for the full system in 8089 position estimations

For future iterations, deriving match certainty from the amount of verified matches could serve as an extra verification step in the state estimation pipeline. To that end, Figure 6.4 displays the relationship between the amount of verified matches and the position error. This confirms the decreased amount of outliers proportional to the amount of verified matches.

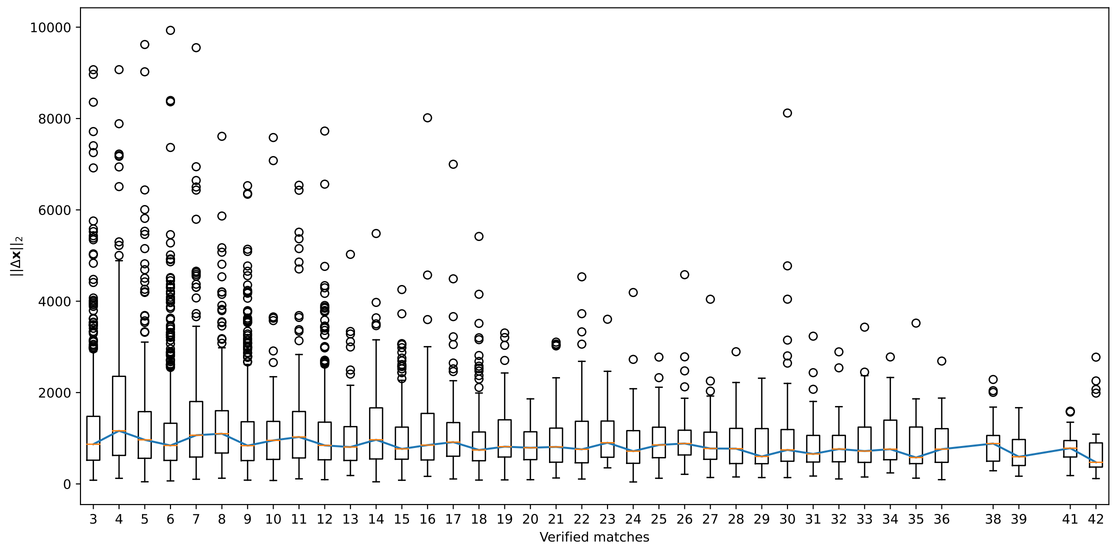


Figure 6.4: Number of verified matches versus position estimation accuracy.

The same box plot was made for both different altitude layers (Figure 6.5) and sun angles (Figure 6.6).

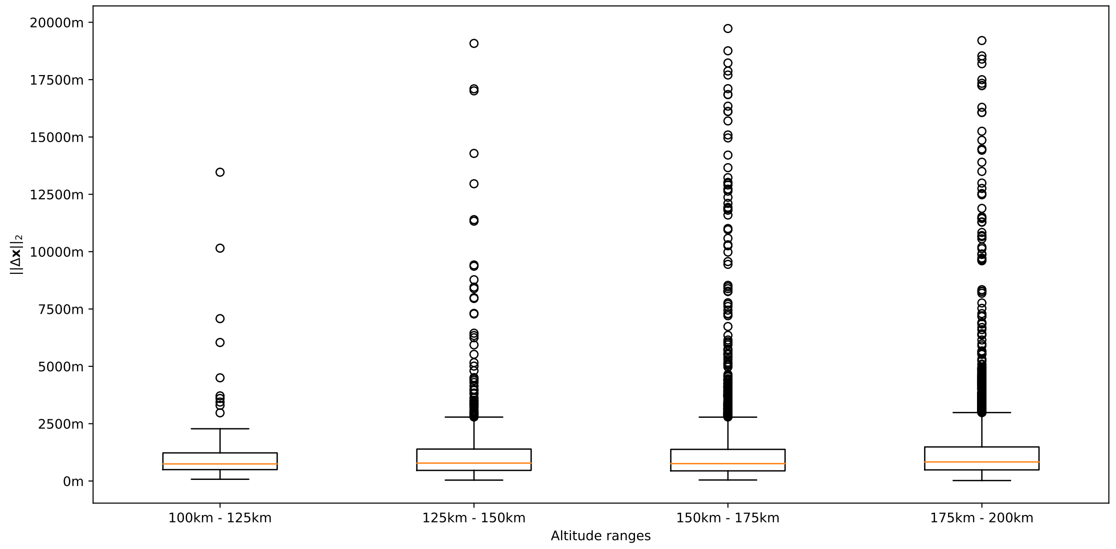


Figure 6.5: Altitude ranges vs. total position error.

Altitude range	Lower whisker	Lower quartile	Median	Upper quartile	Upper whisker
100km - 125km	76m	497m	745m	1226m	2278m
125km - 150km	38m	463m	781m	1393m	2787m
150km - 175km	44m	442m	759m	1380m	2784m
175km - 200km	21m	485m	831m	1486m	2985m

Table 6.2: Error statistics for varying altitude layers.

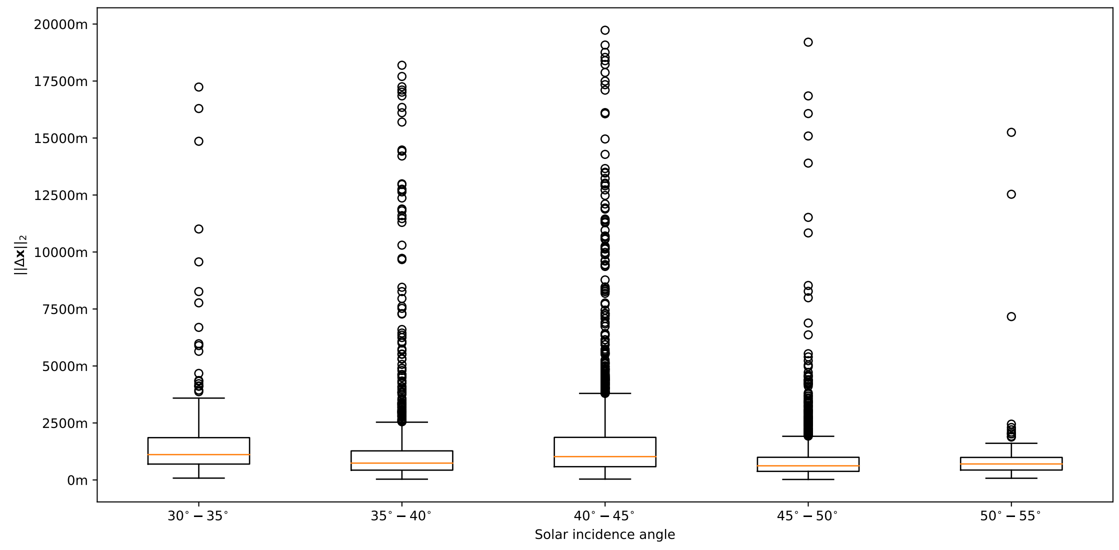


Figure 6.6: Total position error statistics as a boxplot split into sun angle groups.

Sun angle	Lower whisker	Lower quartile	Median	Upper quartile	Upper whisker
30°–35°	80m	696m	1112m	1856m	3590m
35°–40°	34m	430m	738m	1275m	2534m
40°–45°	37m	582m	1023m	1868m	3796m
45°–50°	21m	376m	620m	994m	1915m
50°–55°	74m	434m	701m	988m	1608m

Table 6.3: Total position error statistics as a function of sun angle.

The results from statistical data yielded from Monte Carlo trials is considered sufficient verification at this stage of development as it assesses the system's capability to initialise state without prior information. It can be concluded that, while match rate is insufficient, the proposed system is analytically sound and has the potential to improve in future iterations. The reason why improvements are deemed possible is because the results in [17], which has been the basis for the proposed system's crater identification and position estimation subsystem, claims to have much higher success rate. This means that the cause of this discrepancy is most likely due to implementation differences. Furthermore, the assembled TRN pipeline using the novel CDA introduced in this document together with a projective invariant-based crater pattern matching technique for ego-position estimation is a valid approach.

6.2. Trajectory Performance

To assess a real-world application of the proposed system the system is tested by simulating camera input generated along a pre-defined Kepler orbit above the Moon. Contrary to the instantaneous performance tests, previous measurements can be used to form more accurate state estimations. The state estimation filter used is an Extended Kalman Filter (EKF) with the following characteristics [39]:

$$X_k = \begin{bmatrix} \boldsymbol{x}_B \\ \boldsymbol{v}_B \end{bmatrix}_k = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}_k \tag{6.1}$$

$$\hat{X}_k^- = f(\hat{X}_{k-1}^-) \tag{6.2}$$
$$P_k^- = A_{k-1}P_{k-1}A_{k-1}^T + Q_{k-1} \tag{6.3}$$
$$K_k = P_k^- H_k^T (H_k P_k^- H^T + R_k)^{-1} \tag{6.4}$$
$$\hat{X}_k = \hat{X}_k^- + K_k[z_k - h(\hat{X}_k^-)] \tag{6.5}$$
$$P_k = [I - K_k H_k]P_k^-[I - K_k H_k]^T + K_k R_k K_k^T \tag{6.6}$$

with state transition matrix A defined as

$$A_{k-1}(\hat{X}_{k-1}^+) = I + F_{k-1}(\hat{X}_{k-1}^+)\Delta t, \tag{6.7}$$

with

$$F_{k-1}(\hat{X}_{k-1}^-) = \frac{\partial f}{\partial X}\bigg|_{\hat{X}_{k-1}^-} = \begin{bmatrix} & \mathbf{0}_{3\times3} & & \boldsymbol{I}_{3\times3} \\ \frac{\mu}{||\boldsymbol{x}_B||^5}(3x^2 - ||\boldsymbol{x}_B||^2) & \frac{\mu}{||\boldsymbol{x}_B||^5}(3xy) & \frac{\mu}{||\boldsymbol{x}_B||^5}(3xz) & \\ \frac{\mu}{||\boldsymbol{x}_B||^5}(3xy) & \frac{\mu}{||\boldsymbol{x}_B||^5}(3y^2 - ||\boldsymbol{x}_B||^2) & \frac{\mu}{||\boldsymbol{x}_B||^5}(3yz) & \mathbf{0}_{3\times3} \\ \frac{\mu}{||\boldsymbol{x}_B||^5}(3xz) & \frac{\mu}{||\boldsymbol{x}_B||^5}(3yz) & \frac{\mu}{||\boldsymbol{x}_B||^5}(3z^2 - ||\boldsymbol{x}_B||^2) & \end{bmatrix}_k, \tag{6.8}$$

and measurement matrix H as:

$$H_k = \begin{bmatrix} \boldsymbol{I}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix}, \tag{6.9}$$

with process noise Q_k and measurement noise R given in section A.4. This filter was merely defined to assess state estimation performance given only the updates provided by the proposed system. However, it is assumed that future iterations of this system would incorporate measurement updates from other navigation instruments as well. This test is essentially a worst-case scenario, as no other measurements are included to achieve more accurate state knowledge. The implementation was made using FilterPy [3] for building the EKF and poliastro [72] for ground-truth values of a Lunar orbit.

This test aims to verify **CID-PR-02.3** and **CID-PR-02.4**, which both relate to the system's performance given multiple measurements in time. The results above indicate that it may be possible that the match rate will not be sufficient in every orbit scenario to achieve accurate state estimations, hence only a scenario with sufficient measurement updates will be taken into account to be able to solely assess the system's performance over time. The results of this test will serve as a validation of the full system's performance requirements, as it is tested in a configuration that resembles a real-world application. To achieve this, a nearly circular orbit at roughly 260 [km] above the same patch of Lunar surface given in Table 6.1 is taken. It needs to be stated that there may be situations in which the surface in view of the camera does not contain any or enough craters to be able to query the database with, in which case the state error will increase. Because it is not in the interest of seeing the performance of the proposed system in a trajectory without updates, a trajectory was chosen with enough visible craters.

6.2.1. Results

The system indeed appears to function as would be expected given the state estimation error statistics from the previous section. Even with an initial state estimation degraded as far as 500 [km] (positional) and 800 [m/s] (velocity) is salvageable through the use of the proposed system. Figure 6.7 displays the convergence of the state estimation functioning as expected even in the presence of noisy measurements.

The lowest observed state estimation error observed in this test was 160 [m], which would allow future iterations of the proposed system to incorporate crater pattern matching methods requiring approximate state information (e.g. LunaNet's RANSAC approach [24]), providing additional measurements. To identify the biggest source of state error, the position error is split along-track, cross-track, and radial components that are defined by the velocity components of the ground-truth. The dominant error factor is - as hypothesised and

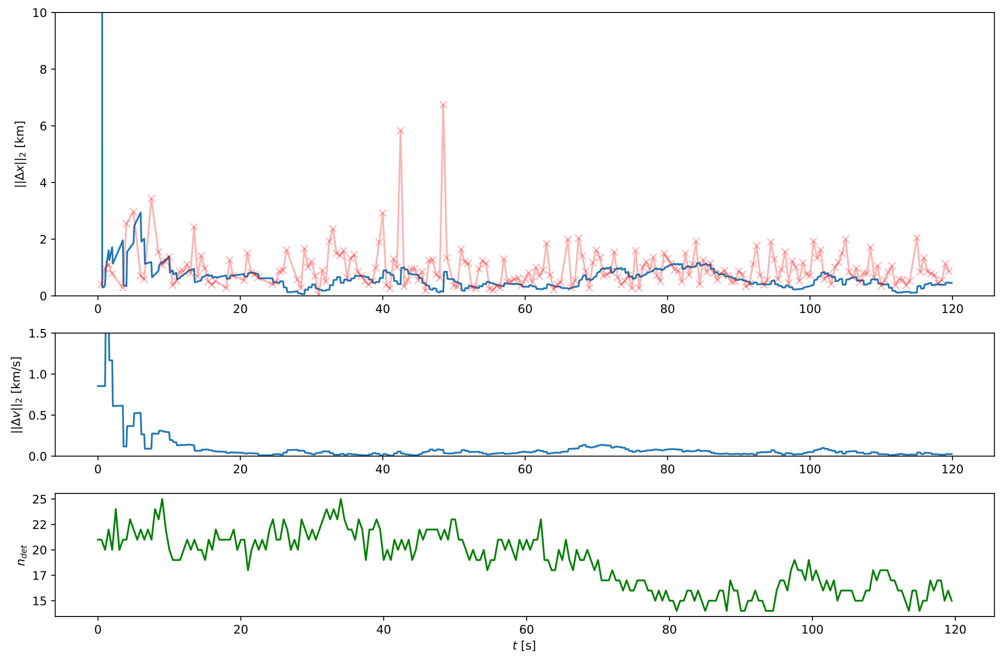


Figure 6.7: Positional state estimations over time with an initial state error of 500 [km]. The top figure shows the absolute positional error, the middle plot shows velocity error, and the lower plot shows the amount of detected craters at each instant. The red crosses in the top plot indicate measurement update errors.

seen in chapter 5 and subsection 6.1.1, Figure 6.8 - the radial component. Future research can focus on assessing a setup with the inclusion of a navigation instrument capable of more accurate altitude measurements.

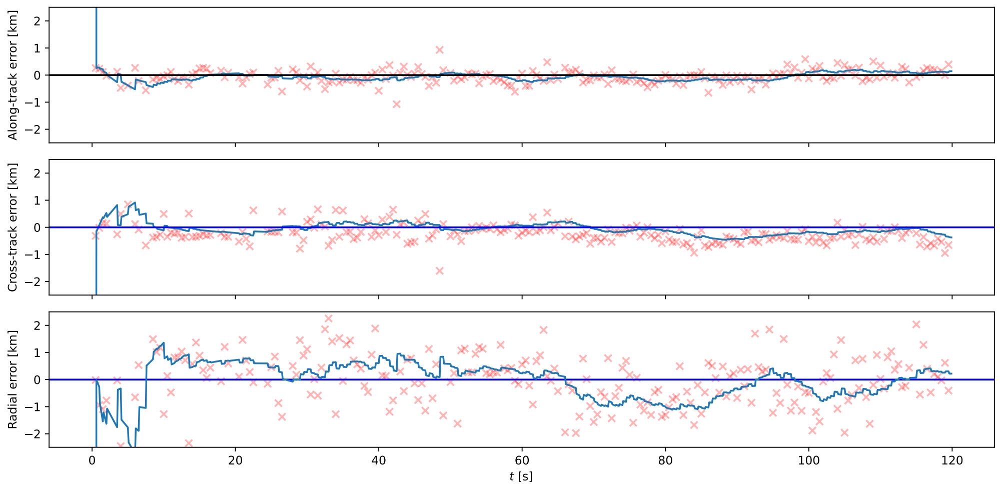


Figure 6.8: The positional error from the top plot in Figure 6.7 split into along-track, cross-track, and radial error. The red crosses indicate measurement errors split into their respective components.

It can be concluded that the proposed system is capable of working in a dynamic environment, and verifying that low match rates are an inconclusive metric for assessing performance in a representative situation. This test demonstrates the combination of a novel CDA combined with an augmented version of the crater pattern matching technique introduced by [17] into a single position estimation pipeline featuring an EKF functioning as hypothesised.

7 Verification & Validation

To identify whether the system merits further research, a full verification of the requirements set in chapter 2 is given. After this, a validation plan is outlined for future work.

7.1. Verification

To assess compliance with the set requirements, an overview is given in a requirements verification matrix (see Table 7.1).

Requirements Verification Matrix				
Top-level requirements				
Functional Requirements				
FR-01: The system shall be able to deliver absolute position estimations from attitude information and camera input alone.				
FR-02: The system shall utilise AI techniques.				
FR-03: The system shall be used for navigating around the Moon.				
FR-04: The system shall comprise all elements required for it to be reproduced by any user that requires it.				
FR-05: The system shall be executable entirely on-board a spacecraft.				
FR-07: The system shall function for any Lunar approach path.				
Performance requirements				
PR-01: The system shall achieve a maximum absolute state error of less than 500 [m] in a Lunar orbit below 500 [km].				
PR-02: The system shall deliver state updates at a frequency of at least TBD [Hz].				
Operational requirements				
OR-01: The system shall be capable of functioning above any cratered region of the Moon without hardware changes.				
OR-02: The system shall fulfil its task autonomously.				
OR-03: The system shall be a self-contained instrument that is capable of handling all necessary steps to transform optical input and attitude information into absolute state knowledge.				
Environmental requirements				
ER-01: The system shall be capable of operating in a lunar orbit environment.				
ER-01.1: The system shall be capable of operating under a SEE error rate of TBD.				
ER-01.2: The system shall be capable of operating with a TID of up to TDB Gy.				
ER-02: The system shall be capable of operating in temperatures ranging from [TBD] to [TBD] [$\circ$ Celsius].				
Component-level requirements				
Crater detection subsystem				
Functional requirements	**Verification status**		**Method**	**Comments**
CDT-FR-01.1: The crater detection subsystem shall be able to detect unobscured craters from a position in orbit above the Lunar surface.	Verified		Camera simulation	Figure 4.4, 4.5, 4.6
CDT-FR-01.2: The crater detection subsystem shall be able to discern individual instanced crater rims.	Verified		Camera simulation	Figure 4.4, 4.5, 4.6
CDT-FR-01.3: The crater detection subsystem shall be able to convert a detected crater rim into parameterised ellipse values.	Verified		Analysis	Equation 4.5, 4.6, 4.7

CDT-FR-02.1: The crater detection subsystem shall utilise a hardware accelerator.	Not verified			HIL test	Out of scope
CDT-FR-03.1: The crater detection subsystem shall be able to generate a set of detected crater rims which comprises parameterised ellipses describing the crater rims in the image plane.	Verified			Camera simulation	Figure 4.4, 4.5, 4.6
CDT-FR-03.2: The crater detection subsystem shall be able to communicate its state to the crater identification subsystem and the GNC subsystem.	Not verified			Simulation HIL test	Out of scope
CDT-FR-03.3: The crater detection subsystem shall be able to communicate detection confidence values to the matching subsystem.	Verified			Analysis	
Performance requirements	**Verification status**	**Target value**	**Tested value**	**Method**	**Comments**
CDT-PR-01.1: The crater detection subsystem shall be able to detect craters with attitudes of up to 30 [degrees] away from nadir-pointing attitude.	Verified	30 [degrees]	45 [degrees]	Camera simulation	Figure 4.5
CDT-PR-01.2: The crater detection subsystem shall be able to detect craters in sunlit conditions with varying sun angles.	Verified	0 to 90 [degrees]	0 to 90 [degrees]	Camera simulation	Figure 4.6, Table 6.3
CDT-PR-01.4: The crater detection subsystem shall be capable of detecting at least 3 craters per given camera input.	Verified	3 [craters]	56 [craters]	Camera simulation	Figure 4.4, 4.5, 4.6
CDT-PR-01.5: The crater detection subsystem shall be able to detect craters with maximum ellipse axis error of 3 [pixels].	Verified	3 [pixels]	2.6 ± 1.8 [pixels]	Camera simulation	
CDT-PR-01.6: The crater detection subsystem shall be able to detect craters with maximum ellipse location error of 3 [pixels].	Verified	3 [pixels]	0.9 ± 0.6 [pixels]	Camera simulation	
CDT-PR-02.1: The crater detection subsystem shall be able to process input images at a rate of TBD [Hz].	Not verified	TBD [Hz]		HIL test	Out of scope
CDT-PR-02.2: The crater detection subsystem shall be able to process input images with a resolution of at least TDB [pixels].	Verified	256 by 256 to 512 by 512 [pixels]	256 by 256 to 512 by 512 [pixels]	Analysis	
Position estimation subsystem					
Functional requirements	**Verification status**			**Method**	**Comments**
CID-FR-01.1: The system shall be able to match triads of approximately coplanar craters.	Verified			Pseudo-measurements Analysis	
CID-FR-01.3: The system shall be able to match detected craters from varying altitudes of at most 500 [km] above the Lunar surface.	Verified			Pseudo-measurements Analysis	Figure 5.2, Table 5.1
CID-FR-01.4: The system shall be able to match detected craters with off-nadir pointing camera angles.	Verified			Pseudo-measurements Analysis	Figure 4.5, Equation 5.3
CID-FR-01.5: The system shall be able to generate a database specific to a desired mission profile.	Verified			Analysis	Figure 5.4
CID-FR-01.6: The system shall be able to match craters from noisy measurements.	Verified			Pseudo-measurements	Figure 5.7, 5.8, 5.9
CID-FR-02.1: The system shall be capable of deriving ego-position estimations from identified craters and spacecraft attitude alone.	Verified			Pseudo-measurements Analysis	Figure 5.7, 5.8, 5.9
CID-FR-02.2: The system shall be capable of rejecting faulty position estimates.	Verified			Analysis	Figure 5.5
CID-FR-02.3: The system shall be capable of integrating prior state information should it be available to improve accuracy.	Not verified			Pseudo-measurements Analysis	Out of scope

Performance requirements	Verification status	Target value	Tested value	Method	Comments
CID-PR-01.2: The system shall be able to create a database for a cratered region that spans at least 1000 by 1000 [km].	Verified	1000 by 1000 [km]	1500 by 1500 [km]	Pseudo-measurements Camera Simulation	
CID-PR-02.1: The system shall be able to derive instantaneous ego-position estimations with an average accuracy below 1 [km].	Verified	1 [km]	800 [m]	Pseudo-measurements Camera Simulation	subsection 6.1.1, Table 2.1 & 6.2
CID-PR-02.3: The system shall be able to derive ego-position estimation error below 487 [m] when multiple measurements are combined through state estimation filtering.	Verified	487 [m]	160 [m]	Trajectory Simulation	Figure 6.7, 6.8
CID-PR-02.4: The system shall be able to function with an initial state uncertainty of at most 500 [km].	Verified	500 [km]	500 [km]	Trajectory Simulation	Figure 6.7, 6.8

Table 7.1: Requirements verification matrix for the proposed system.

The proposed TRN system complies with most but not all set requirements, substantiating the claim that this technology merits further development. The set requirements aim to assess the proposed system's capacity to provide highly accurate position estimations in challenging - Lost-In-Space (LIS) - situations. It is claimed that the simulation environment made possible through the use of SurRender combined with a custom-built interface has a high degree of similarity to the actual environment in which this system will be deployed. Hence, it is concluded that the required levels of performance have been achieved in a representative environment.

Unfortunately, not all requirements have been verified because of resource constraints. The following explanations for all unverified requirements are given:

CDT-FR-02.1 This requirement can only be verified with a HIL setup. This was attempted by using Intel-Movidius' Neural Compute Stick (NCS) platform which harbours a Myriad X chip purpose-built for mobile NN inferencing. Ultimately, this work was dropped due to difficulties in translating a 'proprietary' DL model to a format that could be deployed onto this device. However, there is enough reason to believe that the model can eventually be deployed given the fact that its architecture closely resembles that of Mask R-CNN [45], which has been successfully deployed onto the Myriad platform[1]. This requirement has a high priority should the technology proceed to TRL 4.

CDT-FR-03.2 This requirement was added with the knowledge that the system's adaptivity relies on its ability to handle all possible situations that may arise during operations. Before this requirement could be explored further however, the system's main functionality (position estimation) needed to be verified. Given that this is the case, further development has to point out how a system that is reliant on AI can effectively communicate (error) states.

CDT-PR-02.1 Similar to **CDT-FR-02.1**. Once a HIL setup is created, the system's throughput can be assessed.

CID-FR-02.3 Although not necessary for the system to function, it is hypothesised that the reliability, accuracy, and throughput of the system can be increased by fulfilling this requirement. This requirement has not been verified due to time constraints.

The viability of the technology being advanced is related to the technical risk and effort required before the full system can be validated. Knowing the challenges left open before the full system can be validated, it is concluded that given enough resources this technology can reach TRL 4 in less than a year of work. The effort will mainly focus on repurposing the demonstrator made for this project as a functional and performance test platform with which an embedded version (engineering model) can be verified to work. Modern software engineering practices such as automated testing (using e.g. pytest[2]) can be used to smooth the development

[1]`https://docs.openvinotoolkit.org`
[2]`https://pytest.org`

process. In summary, there is a clear way forward for the proposed system, and current verified requirements support the claim that the system can be fully validated in a more mature development phase.

7.2. Validation Plan

The value of this research lies in its capacity to demonstrate the performance of a TRN system using region-based object detection in tandem with a projective invariant-based crater pattern matching algorithm. To this end, the subsystems that have been developed for the demonstrator have both been verified to work independently, as well as in a full end-to-end system. In fact, the end-to-end system outperforms the pseudo-measurements used for testing the position estimation subsystem.

The identified need for a highly accurate vision-based navigation solution incorporating AI has been reflected in the identified system- and component-level requirements. But, as mentioned in section 2.4, the full system cannot be validated as it requires a more mature iteration of the system along with a higher fidelity testing environment, ultimately resulting in in-flight tests. However the concept has been demonstrated to work through analysis and simulation, justifying further development in the form of an engineering model.

Once an engineering model has been built and when it has been established that all components work together (TRL 4), they can be tested to work in a simulated environment. A representative environment can be constructed by building a to-scale mock up of the lunar surface such as [86]. Another option is to use the same methodology as this project (scene rendering) but instead of directly feeding raw image data to the system, a high resolution image can be projected in front of the camera module to fully verify the systems functionality and performance. The latter option has the same benefits as this project's verification environment: high flexibility and realism. Using such a setup, TRL 5 & 6 iterations can be verified.

The next step (TRL 7) is to verify the system in a space environment, meaning the system is (almost) at fully operational capability. The penultimate verification step is to deploy the system in a demonstration during an actual mission, however it cannot be a mission-critical component at that time, since it is not yet flight proven (TRL 8). If successful, the system can then be used as mission-critical component, meaning it can be labeled flight proven (TRL 9).

8 Conclusions & Recommendations

This chapter contains the concluding remarks on the research that is presented in this document. The outcome presents the answers to the posed research questions, after which recommendations are made for future work.

8.1. Outcome

The main research question was formulated to guide the research to assess the performance characteristics of a TRN system that combines region-based object detection with a projective-invariant-based crater pattern matching technique. This research achieved all three set goals, each of which providing unique results within the domain of TRN and AI.

What requirements should a dataset meant for developing, training, and verifying a Terrain-Relative Navigation system fulfil?

A dataset generator for training and verifying a TRN demonstrator needed to be able to generate physically accurate scenes with high flexibility. Instead of using static camera images or maps, simulation software meant for space exploration development called SurRender was used in tandem with a robust data generation interface that defined all required attributes and methods for scene generation. This meant creating a solid definition of the coordinate system, camera definition, crater projection pipeline, relative planetary positions, and finally the simulated camera output. Using its inherent flexibility, a virtually limitless amount of data can be generated for the purpose of developing, training, and verifying a TRN algorithm.

The creation of the data generation pipeline was further justified during the development and training of the CDA, where it proved capable of generating training data in the right format for a region-based object detection model. Further assessing the system for its ability to work under varying attitude and sun angles was made realistic and easier by the interfaces provided by the data generation class. Similarly, during the development of the position estimation algorithm the pipeline aided in verifying the functionality and performance through pseudo-measurement test. For verifying the end-to-end system a new validation set of 50000 unique images along with ground truth positions was made. This meant that enough results were generated for a statistical analysis into the performance of the full system. Finally, the pipeline was used in tandem with a simulated Kepler orbit, generating simulated camera input from position along the orbit.

This document described a novel method to generate data for training and testing a modern DL model, meaning that its use is not limited to the development of the specific system proposed in this paper. Furthermore, the flexibility and realism introduced by the SurRender software package is deemed to be an improvement for verifying TRN systems compared to static data. Its capacity to use DEM and texture data dynamically with state-of-the-art rendering techniques results in a more representative environment of a real-world situation.

Is a Crater Detection Algorithm using region-based object detection sufficiently robust for a Terrain-Relative Navigation system?

Region-based object detection models have been used before in CDAs, however they have not been developed to output ellipses fit to crater rims in the image plane before. This is a necessity considering the chosen type of crater pattern matching technique requires crater shape information.

This research presents a novel CDA method based on Ellipse R-CNN, which has the capacity to detect instanced crater shapes directly. Instanced detection has the distinct benefit of removing the need for sensitive post-processing steps like those required if semantic segmentation models such as DeepMoon are used. Furthermore, directly predicting ellipse shape parameters is computationally more efficient compared to mask generation and allows the model self-engineer the logic that replaces the post-processing steps through training. This property was used to train the model to minimise the same metric that determined a good ellipse fit in the crater pattern matching algorithm - a novelty in itself. Essentially this proves the effectiveness of developing both subsystems to serve the same end-goal instead of reusing CDAs developed for other purposes, such as crater counting. Furthermore, the system proved robust against changes in spacecraft attitude and exogenous factors such as sunlight and terrain types. Faulty detections are still present, but considering the complex nature of the input data this is to be expected. Putting this result into the context of TRN ameliorates this problem somewhat, since a certain fraction of misfitted ellipses is acceptable as long as position estimations can be made. There is room for further performance improvements, however the proposed CDA's functionality is robust enough for TRN. In conclusion, given the complexity of the input data, it is claimed that a CDA using region-based detection is a valid choice for a TRN system.

How does an absolute position estimation algorithm perform in tandem with a Crater Detection Algorithm using region-based object detection?

Another novelty presented by this research is the combination of a projective-invariant-based crater pattern matching algorithm with position estimation capability in tandem with a purpose-built CDA. The applied techniques for solving the LIS position estimation problem are relatively new at the time of writing, and have therefore not yet been integrated into an end-to-end TRN system. Functionally aligning both the crater detection subsystem with the crater identification and position estimation subsystem allowed for effective simultaneous design. Combined with a representative simulation environment, an integrated TRN solution was developed.

Performance tests have shown that the assembled TRN system meets its functional requirements. The match rate of the complete system in instantaneous performance tests is lacking at only 16%, however the accuracy with which the system is able to derive ego-position from merely camera input and attitude state information is within margin even without the use of a state estimation filter. The results show a median position error of 800 [m], with an unbiased error profile. In conclusion, the combined performance of the novel CDA and the

Is region-based object detection a suitable method for a Lost-In-Space Terrain-Relative Navigation system?

Further tests in which the proposed system combined with an Extended Kalman Filter (EKF) was tested in a simulated orbit around the Moon show that even from a completely uninitialised state - a LIS situation - the state error could be decreased to 160 [m] using only absolute position updates. The results of this implementation show that the system functionally complies with the definition of a LIS TRN, meaning it is capable of initialising state estimation filters often present in GNC subsystems without prior knowledge. The accuracy with which it achieves this is in most cases within the set margins, meaning the detection rate and accuracy is good enough for navigation algorithms. It is concluded that the proposed CDA combined with a crater matching and ego-position algorithm are very well suited for TRN systems, supporting a high degree of autonomy through its ability to work in extreme (LIS) circumstances.

8.2. Recommendations

The proposed system presents a novel TRN algorithm using an AI-based detection model in combination with crater pattern matching. Future development for improving the system should focus on five subjects: data generation improvements, better integration of a state estimation filter, database improvements, and creating an engineering model.

- The data generation pipeline as described in this document is a flexible method of generating limitless data points for the purpose of developing, training, and validating a TRN system. The setup currently is not deemed capable of returning representative images at low altitudes because of the limited resolution in the source DEM. Therefore future work should focus on finding or generating a higher fidelity dataset for simulating the system's performance at lower proximity to the Lunar surface. Furthermore, being able to generate pseudo-measurements such as star tracker and IMU output in real-time is a valid addition to the data generation class, resulting in a testing environment that resembles reality even closer.

- The proposed CDA currently only accepts a single image at a time to form a static output of ellipses fitted to crater rims. This is partially due to the initial focus on instantaneous position estimation performance. However the system's performance is best validated in a trajectory simulation, given the fact that this is more representative of the actual environment in which the system must operate. Using several images over time spent in orbit as input could allow the model to learn relationships between translated versions of a crater (in the image plane) and the spacecraft's state (positional and rotational). A good point to start further research into this is to survey the state-of-the-art of autonomous driving. Essentially, a highly integrated detection model that accepts multiple inputs and which integrates a DL-based state estimation filter based on for example a Long Short Term Memory (LSTM) model could offer great performance benefits.

- The methodology for creating the database presented in this paper is in some part novel, however implementation improvements are required before deploying on an embedded target. Because this system's functionality and performance was demonstrated to work on a workstation, run times were not considered. Effort has been put into creating an efficient querying functionality through a k-d tree, however the library used to achieve this lacked the capability of querying with normalised difference per-axis, resulting in sub-optimal performance. This is a difficult problem, since the database is built beforehand, yet querying needs to apply some sort of weighted distance metric to accurately retrieve the N most alike crater triads. A custom implementation in a compiled language such as C++ may offer the flexibility and speed for real-time querying.

- Then, once the software implementation all set requirements, the system is ready for deployment onto an embedded target. Before this is done however, separate CDA model performance testing can be performed by using a HIL testing setup that only runs the model inference step on a mobile hardware accelerator like a Myriad X. These chips are relatively inexpensive and feature USB connectivity for development purposes. After verifying that the model runs at an acceptable rate on this accelerator, it can be combined with a separate embedded processor for database querying and state estimation filtering to form an end-to-end engineering model for physical tests. These tests can either use similar simulated camera input or use actual camera input from a Lunar surface mock up.

Finally, the reader is encouraged to examine the associated repository containing all the required (Python) code that was written for this project. Cloning this repository is similarly encouraged to promote further research into developing an open source TRN demonstrator.

A Appendix

A.1. Systems Engineering

A.1.1. Full Functional Flow Diagram

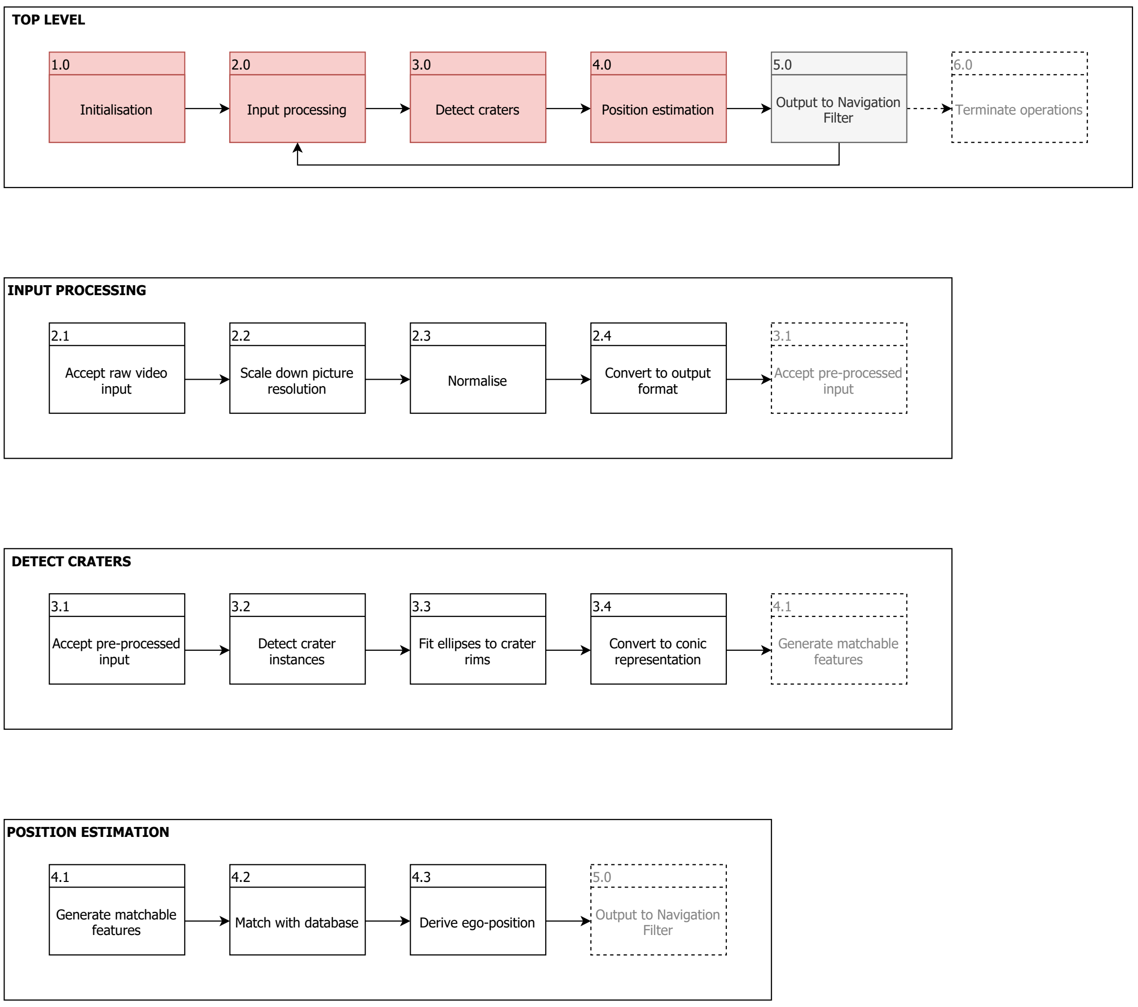

TOP LEVEL
1.0 Initialisation
2.0 Input processing
3.0 Detect craters
4.0 Position estimation
5.0 Output to Navigation Filter
6.0 Terminate operations
INPUT PROCESSING
2.1 Accept raw video input
2.2 Scale down picture resolution
2.3 Normalise
2.4 Convert to output format
3.1 Accept pre-processed input
DETECT CRATERS
3.1 Accept pre-processed input
3.2 Detect crater instances
3.3 Fit ellipses to crater rims
3.4 Convert to conic representation
4.1 Generate matchable features
POSITION ESTIMATION
4.1 Generate matchable features
4.2 Match with database
4.3 Derive ego-position
5.0 Output to Navigation Filter

Figure A.1: Full functional flow diagram for proposed TRN system.

A.2. Detection Model

A.2.1. Backbone First Convolutional Layer Weights

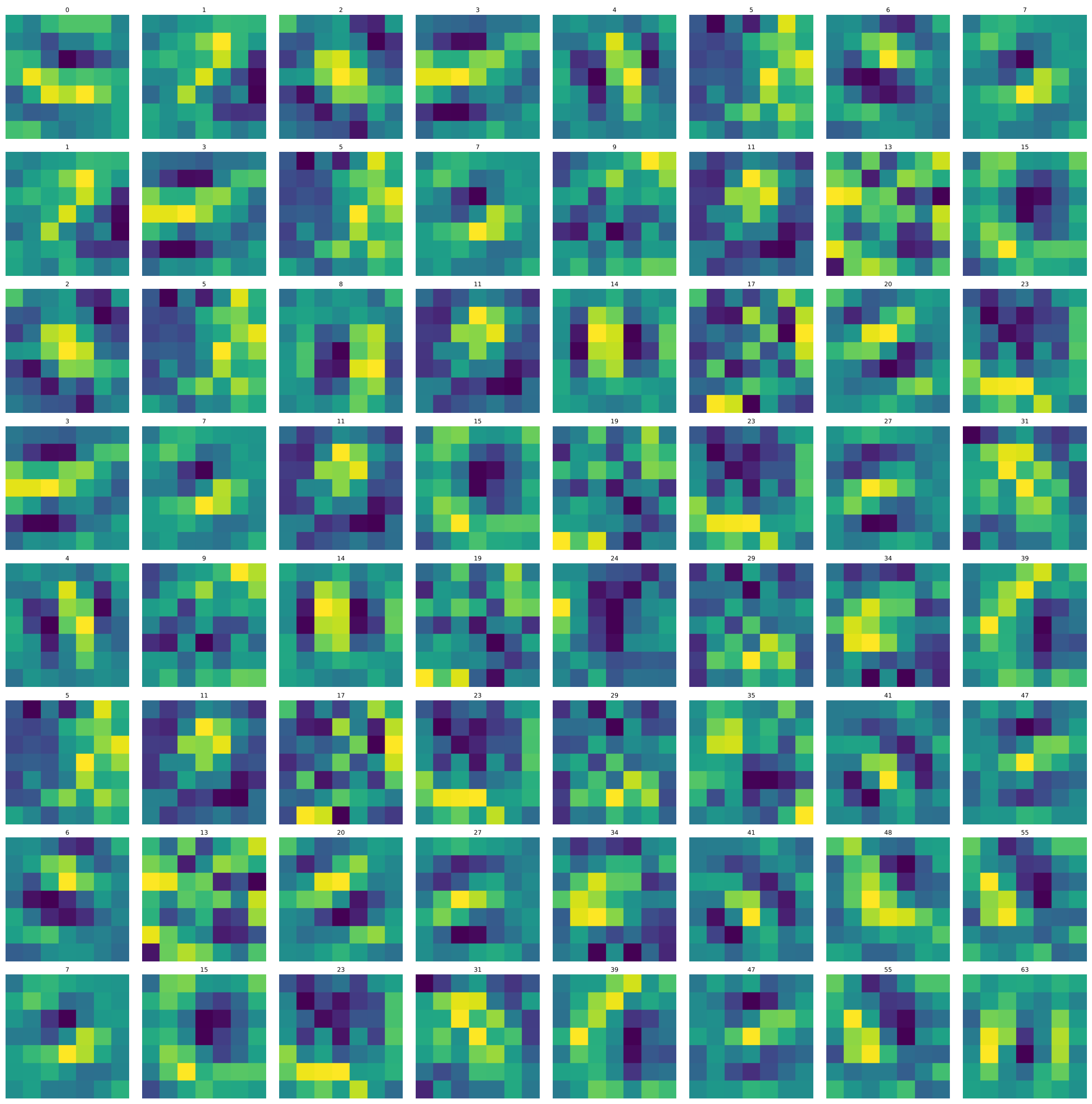


Figure A.2: Visualisation of the weights for all convolutions for the first layer of a trained crater detector using a ResNet50 backbone.

A.2.2. Backbone First Convolutional Layer Output

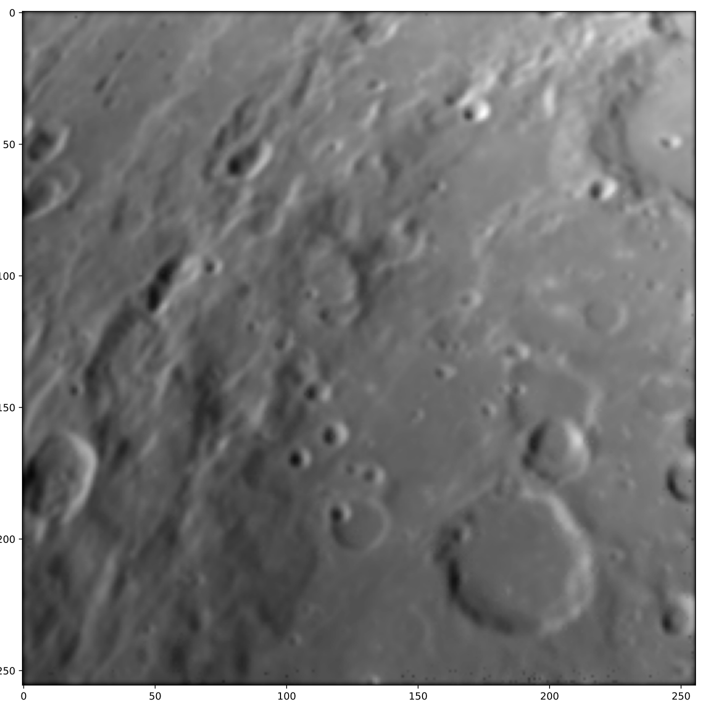


Figure A.3: Example input for Figure A.4.

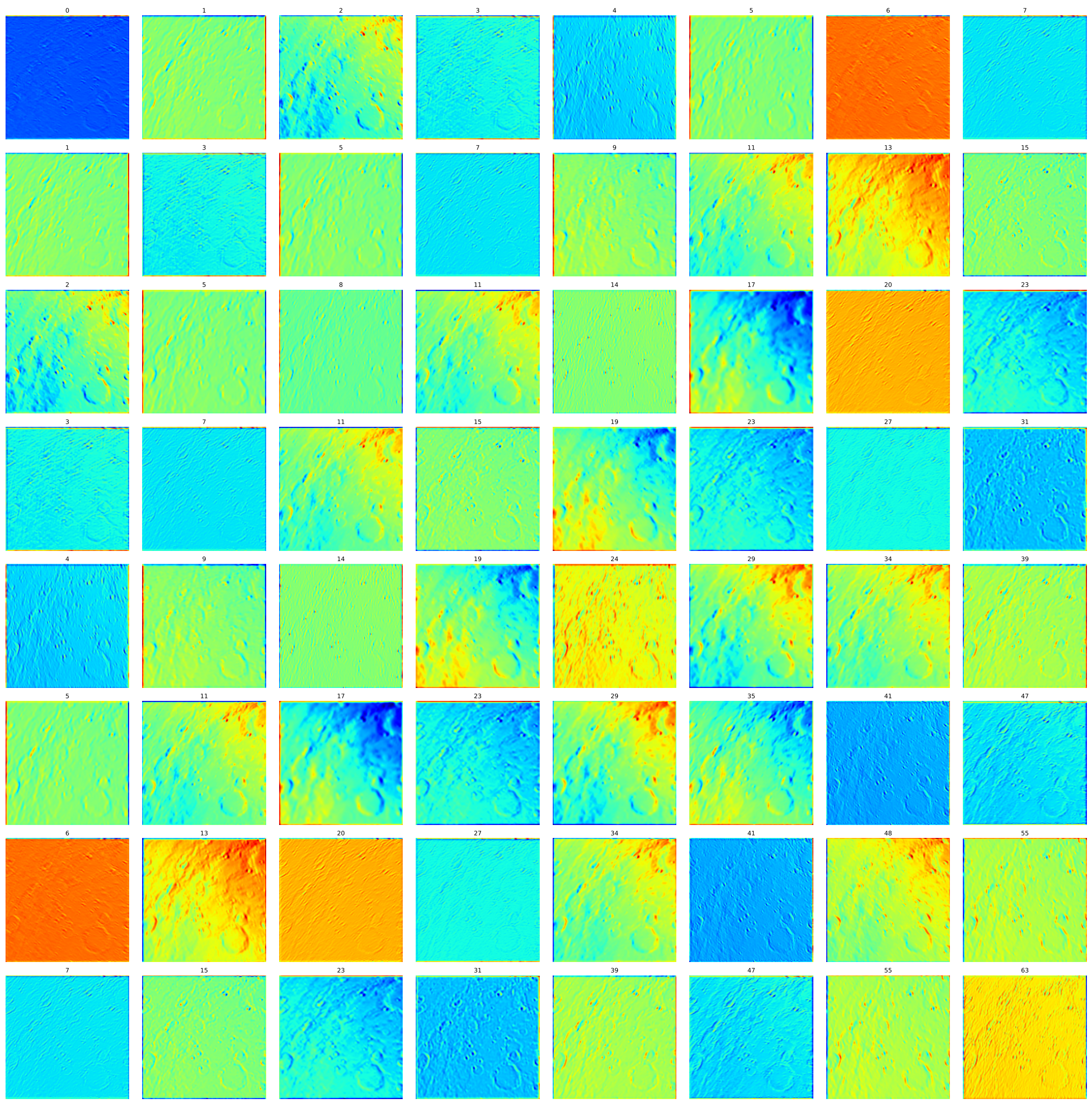


Figure A.4: Visualisation the output of the first layer of a trained crater detector with a ResNet50 backbone on an example input image.

A.2.3. Kullback-Leibler Divergence

$$d_{KL}(\mathcal{N}_i \| \mathcal{N}_i^*) = \frac{1}{2}\left(\mathrm{Tr}(\textstyle\sum_i^* \sum_i) + (\boldsymbol{m}_i^* - \boldsymbol{m}_i)^T \textstyle\sum_i^* (\boldsymbol{m}_i^* - \boldsymbol{m}_i) - 2 + \ln\left(\frac{|\sum_i^*|}{|\sum_i|}\right)\right) \tag{A.1}$$

A.3. Coplanar Invariants

A.3.1. Derivation

Source: p. 52, 53 from [17].

$$|\lambda \boldsymbol{A}_i + \mu \boldsymbol{A}_j + \sigma \boldsymbol{A}_k| = \Theta_1\lambda^3 + \Theta_2\lambda^2\mu + \Theta_3\lambda\mu^2 + \Theta_4\mu^3 + \Theta_5\lambda^2\sigma + \Theta_6\lambda\sigma^2 + \Theta_7\sigma^3 + \Theta_8\mu^2\sigma + \Theta_9\mu\sigma^2 + \Theta_{10}\lambda\mu\sigma, \tag{A.2}$$

$$\Theta_1 = |\boldsymbol{A}_i| \tag{A.3}$$

$$\Theta_4 = |\boldsymbol{A}_j| \tag{A.4}$$

$$\Theta_7 = |\boldsymbol{A}_k| \tag{A.5}$$

and

$$\Theta_2 = \Theta_1 \mathrm{Tr}[\boldsymbol{A}_i^{-1}\boldsymbol{A}_j], \qquad \Theta_3 = \Theta_4 \mathrm{Tr}[\boldsymbol{A}_j^{-1}\boldsymbol{A}_i], \tag{A.6}$$

$$\Theta_5 = \Theta_1 \mathrm{Tr}[\boldsymbol{A}_i^{-1}\boldsymbol{A}_k], \qquad \Theta_6 = \Theta_7 \mathrm{Tr}[\boldsymbol{A}_k^{-1}\boldsymbol{A}_i], \tag{A.7}$$

$$\Theta_8 = \Theta_4 \mathrm{Tr}[\boldsymbol{A}_j^{-1}\boldsymbol{A}_k], \qquad \Theta_9 = \Theta_7 \mathrm{Tr}[\boldsymbol{A}_k^{-1}\boldsymbol{A}_j], \tag{A.8}$$

with Θ_{10} being a coefficient that depends on all three conics, therefore being entirely unique to a single triad given by

$$\Theta_{10} = \frac{1}{2} Tr[(\boldsymbol{A}_j + \boldsymbol{A}_k)^* - (\boldsymbol{A}_j - \boldsymbol{A}_k)^*]\boldsymbol{A}_i. \tag{A.9}$$

Setting

$$|\boldsymbol{A}_i| = |\boldsymbol{A}_j| = |\boldsymbol{A}_k| = 1, \tag{A.10}$$

results in 7 coplanar invariants for a crater triad

$$I_{ij} = Tr[\boldsymbol{A}_i^{-1}\boldsymbol{A}_j] \tag{A.11}$$

$$I_{ji} = Tr[\boldsymbol{A}_j^{-1}\boldsymbol{A}_i] \tag{A.12}$$

$$I_{ik} = Tr[\boldsymbol{A}_i^{-1}\boldsymbol{A}_k] \tag{A.13}$$

$$I_{ki} = Tr[\boldsymbol{A}_k^{-1}\boldsymbol{A}_i] \tag{A.14}$$

$$I_{jk} = Tr[\boldsymbol{A}_j^{-1}\boldsymbol{A}_k] \tag{A.15}$$

$$I_{kj} = Tr[\boldsymbol{A}_k^{-1}\boldsymbol{A}_j] \tag{A.16}$$

$$I_{ijk} = Tr[(\boldsymbol{A}_j + \boldsymbol{A}_k)^* - (\boldsymbol{A}_j - \boldsymbol{A}_k)^*]\boldsymbol{A}_i \tag{A.17}$$

A.4. Extended Kalman Filter Values

$$Q = \begin{bmatrix} 1.25e-08 & 0.00e+00 & 0.00e+00 & 2.50e-07 & 0.00e+00 & 0.00e+00 \\ 0.00e+00 & 1.25e-08 & 0.00e+00 & 0.00e+00 & 2.50e-07 & 0.00e+00 \\ 0.00e+00 & 0.00e+00 & 1.25e-08 & 0.00e+00 & 0.00e+00 & 2.50e-07 \\ 2.50e-07 & 0.00e+00 & 0.00e+00 & 5.00e-06 & 0.00e+00 & 0.00e+00 \\ 0.00e+00 & 2.50e-07 & 0.00e+00 & 0.00e+00 & 5.00e-06 & 0.00e+00 \\ 0.00e+00 & 0.00e+00 & 2.50e-07 & 0.00e+00 & 0.00e+00 & 5.00e-06 \end{bmatrix} \tag{A.18}$$

$$R = \begin{bmatrix} 0.1 & 0. & 0. \\ 0. & 0.1 & 0. \\ 0. & 0. & 0.1 \end{bmatrix} \tag{A.19}$$

A.5. Code Repository

Because this project required the creation of a significant amount of code (over 4500 lines of Python), it was decided that the best way to share this is through a public online repository hosted on GitHub. This allows the reader to search through the code, and clone it for their own use. The code is shipped with an MIT license[1]. Click (if reading with a PDF viewer) or scan the QR-code given in Figure A.5 to access it.

Figure A.5: GitHub Repository QR code.

A.6. Workstation Specifications

Component	**Name**
Central Processing Unit	AMD Ryzen 7 5800X, 8C16T
Graphics Processing Unit	NVIDIA RTX 3080
RAM	32 GB 3600 MHz, C18
SSD	1TB Samsung 980 Pro

Table A.1: Workstation specifications.

[1] `https://opensource.org/licenses/MIT`

Bibliography

[1] How Neural Networks process input data - AI In Plain English - Medium. URL https://medium.com/ai-in-plain-english/my-notes-on-neural-networks-adf3e49657f8.

[2] LROC WMS Image Map. URL http://wms.lroc.asu.edu/lroc.

[3] rlabbe/filterpy: Python Kalman filtering and optimal estimation library. Implements Kalman filter, particle filter, Extended Kalman filter, Unscented Kalman filter, g-h (alpha-beta), least squares, H Infinity, smoothers, and more. Has companion book 'Kalman and Bayesian Filters in Python'. URL https://github.com/rlabbe/filterpy.

[4] Robust linear model estimation using RANSAC — scikit-learn 0.24.2 documentation. URL https://scikit-learn.org/stable/auto_examples/linear_model/plot_ransac.html.

[5] Tycho Crater on the Moon (Labeled) | NASA Solar System Exploration. URL https://solarsystem.nasa.gov/resources/2264/tycho-crater-on-the-moon-labeled/.

[6] Vega VV16 launched, carrying PhiSat-1 with Ubotica AI technology on board – Ubotica Technologies. URL https://ubotica.com/vega-vv16-launch-carrying-phisat-1-with-ubotica-ai-technology-on-board/.

[7] Moon LRO LOLA DEM 118m v1 | USGS Astrogeology Science Center, 1 2014. URL https://astrogeology.usgs.gov/search/map/Moon/LRO/LOLA/Lunar_LRO_LOLA_Global_LDEM_118m_Mar2014.

[8] David Arnas, Márcio A.A. Fialho, and Daniele Mortari. Fast and robust kernel generators for star trackers. *Acta Astronautica*, 134(August 2016):291–302, 5 2017. ISSN 00945765. doi: 10.1016/j.actaastro.2017.02.016. URL http://dx.doi.org/10.1016/j.actaastro.2017.02.016https://linkinghub.elsevier.com/retrieve/pii/S0094576516308232.

[9] M. K. Barker, E. Mazarico, G. A. Neumann, M. T. Zuber, J. Haruyama, and D. E. Smith. A new lunar digital elevation model from the Lunar Orbiter Laser Altimeter and SELENE Terrain Camera. *Icarus*, 273: 346–355, 7 2016. ISSN 10902643. doi: 10.1016/j.icarus.2015.07.039.

[10] Jon Louis Bentley. Multidimensional Binary Search Trees Used for Associative Searching. *Communications of the ACM*, 18(9):509–517, 9 1975. ISSN 15577317. doi: 10.1145/361002.361007. URL https://dl.acm.org/doi/abs/10.1145/361002.361007.

[11] V. Simard Bilodeau, S. Clerc, R. Drai, and J. de Lafontaine. Optical Navigation System for Pin-Point Lunar Landing. *IFAC Proceedings Volumes*, 47(3):10535–10542, 2014. ISSN 14746670. doi: 10.3182/20140824-6-ZA-1003.01693. URL http://dx.doi.org/10.3182/20140824-6-ZA-1003.01693https://linkinghub.elsevier.com/retrieve/pii/S1474667016432878.

[12] T. Brady and J. Schwartz. ALHAT system architecture and operational concept. *IEEE Aerospace Conference Proceedings*, (617), 2007. ISSN 1095323X. doi: 10.1109/AERO.2007.352725.

[13] John Canny. A Computational Approach to Edge Detection. Technical Report 6, 1986.

[14] David Casasent and Michael Saverino. Optical Image Processing For Missile Guidance. In *Proc.SPIE*, volume 0118, 12 1977. doi: 10.1117/12.955668. URL https://doi.org/10.1117/12.955668.

[15] Yang Cheng and James K. Miller. AUTONOMOUS LANDMARK BASED SPACECRAFT NAVIGATION SYSTEM. Technical report, 2003.

[16] John A Christian. A Tutorial on Horizon-Based Optical Navigation and Attitude Determination with Space Imaging Systems. 2021. doi: 10.1109/ACCESS.2021.3051914.

[17] John A. Christian, Harm Derksen, and Ryan Watkins. Lunar Crater Identification in Digital Images. 9 2020. URL `http://arxiv.org/abs/2009.01228`.

[18] Sébastien Clerc, Marc Spigai, and Vincent Simard-Bilodeau. *A crater detection and identification algorithm for autonomous lunar landing*, volume 7. IFAC, 2010. ISBN 9783902661876. doi: 10.3182/20100906-3-it-2019.00091. URL `http://dx.doi.org/10.3182/20100906-3-IT-2019.00091`.

[19] Michael J. Cullinane. Metric axioms and distance. *The Mathematical Gazette*, 95(534):414–419, 11 2011. ISSN 0025-5572. doi: 10.1017/s0025557200003508. URL `https://doi.org/10.1017/S0025557200003508`.

[20] Jesse Davis and Mark Goadrich. The Relationship Between Precision-Recall and ROC Curves. Technical report.

[21] D. M. DeLatte, S. T. Crites, N. Guttenberg, and T. Yairi. Automated crater detection algorithms from a machine learning perspective in the convolutional neural network era. *Advances in Space Research*, 64 (8):1615–1628, 2019. ISSN 18791948. doi: 10.1016/j.asr.2019.07.017. URL `https://doi.org/10.1016/j.asr.2019.07.017`.

[22] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. pages 248–255. Institute of Electrical and Electronics Engineers (IEEE), 3 2010. doi: 10.1109/cvpr.2009.5206848.

[23] Wenbo Dong, Pravakar Roy, Cheng Peng, and Volkan Isler. Ellipse R-CNN: Learning to Infer Elliptical Object from Clustering and Occlusion. *IEEE Transactions on Image Processing*, 30:2193–2206, 1 2020. doi: 10.1109/TIP.2021.3050673. URL `http://arxiv.org/abs/2001.11584http://dx.doi.org/10.1109/TIP.2021.3050673`.

[24] Lena M Downes, Ted J Steiner, and Jonathan P How. Lunar Terrain Relative Navigation Using a Convolutional Neural Network for Visual Crater Detection. In *2020 American Control Conference (ACC)*, volume 2020-July, pages 4448–4453. IEEE, 7 2020. ISBN 978-1-5386-8266-1. doi: 10.23919/ACC45564.2020.9147595. URL `https://ieeexplore.ieee.org/document/9147595/`.

[25] Paul Duteïs, Roland Brochard, Darius Djafari-rouhani, Manuel Sanchez Gestido, Airbus Defence, and European Space Agency. Genevis : Generic Vision Based Navigation for Descent & Landing. *2nd RPI Space Imaging Workshop*, (October):2–3, 2019.

[26] ECSS. ECSS-E-ST-70-11C - Space segment operability. (July), 2008.

[27] Ebrahim Emami, George Bebis, Ara Nefian, and Terry Fong. Automatic Crater Detection Using Convex Grouping and Convolutional Neural Networks. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9475, pages 213–224. 2015. ISBN 9783319278629. doi: 10.1007/978-3-319-27863-6{_}20. URL `http://link.springer.com/10.1007/978-3-319-27863-6_20`.

[28] Ebrahim Emami, Touqeer Ahmad, George Bebis, Ara Nefian, and Terry Fong. Crater Detection Using Unsupervised Algorithms and Convolutional Neural Networks. *IEEE Transactions on Geoscience and Remote Sensing*, 57(8):5373–5383, 8 2019. ISSN 15580644. doi: 10.1109/TGRS.2019.2899122.

[29] ESTEC. Artificial Intelligence Techniques in On-Board Avionics and Software. Technical report, 2019.

[30] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, 6 1981. ISSN 15577317. doi: 10.1145/358669.358692. URL `https://dl.acm.org/doi/abs/10.1145/358669.358692`.

[31] David Forsyth, Joseph L. Mundy, Andrew Zisserman, Chris Coelho, Aaron Heller, and Charles Rothwell. Invariant Descriptors for 3-D Object Recognition and Pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):971–991, 1991. ISSN 01628828. doi: 10.1109/34.99233.

[32] Andrea Frome, Daniel Huber, Ravi Kolluri, Thomas Bülow, and Jitendra Malik. Recognizing objects in range data using regional point descriptors. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3023:224–237, 2004. ISSN 16113349. doi: 10.1007/978-3-540-24672-5{_}18.

[33] Gianluca Furano, European Space Agency, Ireland Kay-obbe Voss, and G S I Helmholtz Centre. Towards the Use of Artificial Intelligence on the Edge in Space Systems : Challenges and Opportunities. (10), 2020.

[34] Ross Girshick. Fast R-CNN. In *2015 IEEE International Conference on Computer Vision (ICCV)*, volume 2015 Inter, pages 1440–1448. IEEE, 12 2015. ISBN 978-1-4673-8391-2. doi: 10.1109/ICCV.2015.169. URL `http://ieeexplore.ieee.org/document/7410526/`.

[35] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014. ISBN 9781479951178. doi: 10.1109/CVPR.2014.81. URL `http://www.cs.berkeley.edu/�rbg/rcnn`.

[36] Joe P. Golden. Terrain Contour Matching (TERCOM): A Cruise Missile Guidance Aid</title>. In *Image Processing for Missile Guidance*, volume 0238, pages 10–18. SPIE, 12 1980. doi: 10.1117/12.959127.

[37] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[38] Patrick Gros and Long Quan. Projective Invariants for Vision. Technical report, 2011. URL `https://hal.inria.fr/inria-00590013`.

[39] Tamer Mekky Ahmed Habib. Simultaneous spacecraft orbit estimation and control based on GPS measurements via extended Kalman filter. *Egyptian Journal of Remote Sensing and Space Science*, 16(1):11–16, 2013. ISSN 20902476. doi: 10.1016/j.ejrs.2012.11.002.

[40] Chad Hanak, Tim Crain, and Robert Bishop. CRATER IDENTIFICATION ALGORITHM FOR THE LOST IN LOW LUNAR ORBIT SCENARIO. *AAS 08-XXX*, 77058, 2009.

[41] Francis Chad Hanak. Lost in Low Lunar Orbit Crater Pattern Detection and Identification. 2009.

[42] William K. Hartmann. History of the terminal cataclysm paradigm: Epistemology of a planetary bombardment that never (?) happened. *Geosciences (Switzerland)*, 9(7):285, 7 2019. ISSN 20763263. doi: 10.3390/geosciences9070285. URL `www.mdpi.com/journal/geosciences`.

[43] Jiang He, Hutao Cui, and Junhua Feng. Edge information based crater detection and matching for lunar exploration. In *2010 International Conference on Intelligent Control and Information Processing*, number PART 2, pages 302–307. IEEE, 8 2010. ISBN 978-1-4244-7047-1. doi: 10.1109/ICICIP.2010.5565284. URL `http://ieeexplore.ieee.org/document/5565284/`.

[44] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-December, pages 770–778. IEEE Computer Society, 12 2016. ISBN 9781467388504. doi: 10.1109/CVPR.2016.90. URL `http://image-net.org/challenges/LSVRC/2015/`.

[45] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):386–397, 2 2020. ISSN 19393539. doi: 10.1109/TPAMI.2018.2844175. URL `https://github.com/`.

[46] James W. Head, Caleb I. Fassett, Seth J. Kadish, David E. Smith, Maria T. Zuber, Gregory A. Neumann, and Erwan Mazarico. Global distribution of large lunar craters: Implications for resurfacing and impactor populations. *Science*, 329(5998):1504–1507, 9 2010. ISSN 00368075. doi: 10.1126/science.1195050. URL `http://science.sciencemag.org/`.

[47] Douglas R. Heisterkamp and Prabir Bhattacharya. Invariants of families of coplanar conics and their applications to object recognition. In *Proceedings - International Conference on Pattern Recognition*, volume 1, pages 677–681. Institute of Electrical and Electronics Engineers Inc., 1996. ISBN 081867282X. doi: 10.1109/ICPR.1996.546110.

[48] David G. Hoag. Apollo - Guidance, Navigation and Control. 1969.

[49] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 1999. ISSN 01628828. doi: 10.1109/34.765655.

[50] Andrew E. Johnson and James F. Montgomery. Overview of Terrain Relative Navigation approaches for precise lunar landing. *IEEE Aerospace Conference Proceedings*, 2008. ISSN 1095323X. doi: 10.1109/AERO.2008.4526302.

[51] Allen Kent, Madeline M. Berry, Fred U. Luehrs, and J. W. Perry. Machine literature searching VIII. Operational criteria for designing information retrieval systems. *American Documentation*, 6(2):93–101, 4 1955. ISSN 0096946X. doi: 10.1002/asi.5090060209.

[52] W. Larson. *Applied space systems engineering*. 2009. URL `https://www.semanticscholar.org/paper/Applied-space-systems-engineering-Larson/7dceac2ac68830780a92298f3d490dd37bb95596`.

[53] George Lentaris, Konstantinos Maragos, Ioannis Stratakos, Lazaros Papadopoulos, Odysseas Papanikolaou, Dimitrios Soudris, Manolis Lourakis, Xenophon Zabulis, David Gonzalez-Arjona, and Gianluca Furano. High-Performance Embedded Computing in Space: Evaluation of Platforms for Vision-Based Navigation. *Journal of Aerospace Information Systems*, 15(4):178–192, 4 2018. ISSN 2327-3097. doi: 10.2514/1.I010555. URL `https://arc.aiaa.org/doi/10.2514/1.I010555`.

[54] Vasileios Leon, George Lentaris, Evangelos Petrongonas, Dimitrios Soudris, Gianluca Furano, and David Moloney. Improving Performance-Power-Programmability in Space Avionics with Edge Devices : VBN on Myriad2 SoC. 99(9), 2020.

[55] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C Lawrence Zitnick, and Piotr Dolí. Microsoft COCO: Common Objects in Context. Technical report.

[56] Jianjun Liu, Xin Ren, Wei Yan, Chunlai Li, He Zhang, Yang Jia, Xingguo Zeng, Wangli Chen, Xingye Gao, Dawei Liu, Xu Tan, Xiaoxia Zhang, Tao Ni, Hongbo Zhang, Wei Zuo, Yan Su, and Weibin Wen. Descent trajectory reconstruction and landing site positioning of Chang'E-4 on the lunar farside. *Nature Communications*, 10(1), 2019. ISSN 20411723. doi: 10.1038/s41467-019-12278-3. URL `http://dx.doi.org/10.1038/s41467-019-12278-3`.

[57] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 11 2004. ISSN 09205691. doi: 10.1023/B:VISI.0000029664.99615.94. URL `https://link.springer.com/article/10.1023/B:VISI.0000029664.99615.94`.

[58] Tingting Lu, Weiduo Hu, Chang Liu, and Daguang Yang. Relative pose estimation of a lander using crater detection and matching. *Optical Engineering*, 55(2):023102, 2 2016. ISSN 0091-3286. doi: 10.1117/1.OE.55.2.023102. URL `http://opticalengineering.spiedigitallibrary.org/article.aspx?doi=10.1117/1.OE.55.2.023102`.

[59] R. Duncan Luce and Albert D. Perry. A method of matrix analysis of group structure. *Psychometrika*, 14(2):95–116, 6 1949. ISSN 00333123. doi: 10.1007/BF02289146. URL `https://link.springer.com/article/10.1007/BF02289146`.

[60] Marco Mammarella, Marcos Avilés Rodrigálvarez, Andrea Pizzichini, and Ana María Sánchez Montero. Advanced Optical Terrain Absolute Navigation for Pinpoint Lunar Landing. In *Advances in Aerospace Guidance, Navigation and Control*, volume 7, pages 419–430. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 9783642198168. doi: 10.1007/978-3-642-19817-5{_}32. URL `http://link.springer.com/10.1007/978-3-642-19817-5_32`.

[61] NAIF. Most Useful SPICELIB Subroutines. (March), 1999. URL https://pirlwww.lpl.arizona.edu/resources/guide/software/SPICE/old_tutorials/mostused.pdf.

[62] Niall O'Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. Deep Learning vs. Traditional Computer Vision. *Advances in Intelligent Systems and Computing*, 943(Cv):128–144, 2020. ISSN 21945365. doi: 10.1007/978-3-030-17795-9{_}10.

[63] Woosang Park, Youeyun Jung, and Hyochoong Bang. Terrain relative navigation for precise lunar landing using crater matching algorithm. *International Conference on Control, Automation and Systems*, 0(Iccas): 582–586, 2016. ISSN 15987833. doi: 10.1109/ICCAS.2016.7832378.

[64] 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.

[65] Pedro Domingos. A Few Useful Things to Know About Machine Learning. *communications of the ACM*, 55(10):79–88, 2012. URL https://dl.acm.org/citation.cfm?id=2347755.

[66] Long Quan and Francoise Veillon. Joint Invariants of a Triplet of Coplanar Conics: Stability and Discriminating Power for Object Recognition. *Computer Vision and Image Understanding*, 70(1):111–119, 4 1998. ISSN 10773142. doi: 10.1006/cviu.1998.0617.

[67] Berjaoui; R. Brochard; J. Lebreton; C. Robin; K. Kanani; G. Jonniaux; A. Masson; N. Despré; A. Scientific image rendering for space scenes with the SurRender software. 2018. ISSN 1098-6596.

[68] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. Technical report. URL https://pjreddie.com/yolo/.

[69] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 6 2017. ISSN 01628828. doi: 10.1109/TPAMI.2016.2577031. URL http://image-net.org/challenges/LSVRC/2015/results.

[70] David Rijlaarsdam, Hamza Yous, Jonathan Byrne, Davide Oddenino, Gianluca Furano, and David Moloney. A Survey of Lost-in-Space Star Identification Algorithms Since 2009. *Sensors*, 20(9):2579, 5 2020. ISSN 1424-8220. doi: 10.3390/s20092579. URL https://www.mdpi.com/1424-8220/20/9/2579.

[71] Stuart J. Robbins. A New Global Database of Lunar Impact Craters &gt;1–2 km: 1. Crater Locations and Sizes, Comparisons With Published Databases, and Global Analysis. *Journal of Geophysical Research: Planets*, 124(4):871–892, 4 2019. ISSN 2169-9097. doi: 10.1029/2018JE005592. URL https://onlinelibrary.wiley.com/doi/abs/10.1029/2018JE005592.

[72] Juan Luis Cano Rodríguez, Jorge Martínez Garrido, Antonio Hidalgo, Shreyas Bapat, Nikita Astrakhantsev, Eleftheria Chatziargyriou, María Eugenia Cruz, Dani, Abhishek Chaurasia, Yash-10, Alberto Lorenzo Márquez, Dhruv Sondhi, Tomek Mrugalski, Emily Selwood, Orestis Ousoultzoglou, Pablo Rodríguez Robles, Greg Lindahl, andrea-carballo, Andrej Rode, Helge Eichhorn, Himanshu Garg, Hrishikesh Goyal, Ian DesJardin, aOrionis, AntoniiaK, Josvth, Priyanshu Rohilla, Angala, Ole Streicher, and Rafael Araujo Lehmkuhl. poliastro/poliastro: poliastro 0.15.0 (Earth edition). 5 2021. doi: 10.5281/ZENODO.4763538. URL https://zenodo.org/record/4763538.

[73] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9351, pages 234–241. 2015. ISBN 9783319245737. doi: 10.1007/978-3-319-24574-4{_}28. URL http://link.springer.com/10.1007/978-3-319-24574-4_28.

[74] Bikash Sabata and J. K. Aggarwal. Estimation of motion from a pair of range images: A review. *CVGIP: Image Understanding*, 54(3):309–324, 11 1991. ISSN 10499660. doi: 10.1016/1049-9660(91)90032-K.

[75] Goran Salamunićcar, Sven Lonarić, Pedro Pina, Loureno Bandeira, and Jos Saraiva. MA130301GT catalogue of Martian impact craters and advanced evaluation of crater detection algorithms using diverse topography and image datasets. *Planetary and Space Science*, 59(1):111–131, 1 2011. ISSN 00320633. doi: 10.1016/j.pss.2010.11.003.

[76] Goran Salamunićcar, Sven Lončarić, and Erwan Mazarico. LU60645GT and MA132843GT catalogues of Lunar and Martian impact craters developed using a Crater Shape-based interpolation crater detection algorithm for topography data. In *Planetary and Space Science*, volume 60, pages 236–247. Pergamon, 1 2012. doi: 10.1016/j.pss.2011.09.003.

[77] Y. Sheikh, S. Khan, M. Shah, and Richard Cannata. Geodetic Alignment of Aerial Video Frames. pages 144–179. 2003. doi: 10.1007/978-1-4615-0459-7{_}7.

[78] Ari Silburt, Mohamad Ali-Dib, Chenchong Zhu, Alan Jackson, Diana Valencia, Yevgeni Kissin, Daniel Tamayo, and Kristen Menou. Lunar crater identification via deep learning. *Icarus*, 317(March):27–38, 1 2019. ISSN 00191035. doi: 10.1016/j.icarus.2018.06.022. URL https://linkinghub.elsevier.com/retrieve/pii/S0019103518301386.

[79] V. Simard Bilodeau, D. Neveu, S. Bruneau-Dbuc, M. Alger, J. de LaFontaine, S. Clerc, and R. Drai. Pinpoint Lunar Landing Navigation using Crater Detection and Matching: Design and Laboratory Validation. In *AIAA Guidance, Navigation, and Control Conference*, number August, pages 1–25, Reston, Virigina, 8 2012. American Institute of Aeronautics and Astronautics. ISBN 978-1-60086-938-9. doi: 10.2514/6.2012-5032. URL http://arc.aiaa.org/doi/10.2514/6.2012-5032.

[80] Karen Simonyan and Andrew Zisserman. VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION. Technical report, 2015. URL http://www.robots.ox.ac.uk/.

[81] David E Smith, Maria T Zuber, Gregory A Neumann, Erwan Mazarico, James Head, Mark H Torrence, and the LOLA Science Team. RESULTS FROM THE LUNAR ORBITER LASER ALTIMETER (LOLA): GLOBAL, HIGH-RESOLUTION TOPOGRAPHIC MAPPING OF THE MOON. doi: 10.1007/s11214-007-9153-y.

[82] TEC-SHS. TECHNOLOGY READINESS LEVELS HANDBOOK FOR SPACE APPLICATIONS. Technical report, 9 2008.

[83] Craig R. Tooley, Martin B. Houghton, Richard S. Saylor, Cathy Peddie, David F. Everett, Charles L. Baker, and Kristina N. Safdie. Lunar reconnaissance orbiter mission and spacecraft design. *Space Science Reviews*, 150(1-4):23–62, 1 2010. ISSN 00386308. doi: 10.1007/s11214-009-9624-4. URL www.nasa.gov.

[84] Nikolas Trawny, Anastasios I. Mourikis, Stergios I. Roumeliotis, Andrew E. Johnson, and James F. Montgomery. Vision-aided inertial navigation for pin-point landing using observations of mapped landmarks. *Journal of Field Robotics*, 24(5):357–378, 5 2007. ISSN 15564959. doi: 10.1002/rob.20189.

[85] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J van der Walt, Matthew Brett, Joshua Wilson, K Jarrod Millman, Nikolay Mayorov, Andrew R J Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E A Quintero, Charles R Harris, Anne M Archibald, Antônio H Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[86] Thomas Voirin, Jeff Delaune, G Le Besnerais, J L Farges, Clément Bourdarias, and Hans Krueger. CHALLENGES OF PINPOINT LANDING FOR PLANETARY EXPLORATION : THE LION ABSOLUTE VISION-BASED NAVIGATION SYSTEM STEP-WISE VALIDATION APPROACH. Technical report.

[87] Hao Wang, Jie Jiang, and Guangjun Zhang. CraterIDNet: An end-to-end fully convolutional neural network for crater detection and identification in remotely sensed planetary images. *Remote Sensing*, 10(7), 2018. ISSN 20724292. doi: 10.3390/rs10071067.

[88] H Wv and G Chen. Suboptimal Kalman Filtering for Linear Systems with Gaussian-Sum Type of Noise. Technical report, 1999.

[89] Meng Yu, Hutao Cui, and Yang Tian. A new approach based on crater detection and matching for visual navigation in planetary landing. *Advances in Space Research*, 53(12):1810–1821, 6 2014. ISSN 02731177. doi: 10.1016/j.asr.2013.04.011. URL `http://dx.doi.org/10.1016/j.asr.2013.04.011https://linkinghub.elsevier.com/retrieve/pii/S0273117713002238`.

[90] Tong Yu and Hong Zhu. Hyper-Parameter Optimization: A Review of Algorithms and Applications. Technical report.