Optical Navigation for Lunar Landing based on Convolutional Neural Network Crater Detector

Stefano Silvestrini[a], Margherita Piccinin[a], Giovanni Zanotti[a], Andrea Brandonisio[a], Ilaria Bloise[b], Lorenzo Feruglio[b], Paolo Lunghi[a], Michèle Lavagna[a], Mattia Varile[b]

[a]*Politecnico di Milano – Department of Aerospace Science and Technology, via La Masa, 34, Milano, 20134, MI, Italy*
[b]*AIKO S.R.L., via Dei Mille, 22, Torino, 10123, TO, Italy*

Abstract

Traditional vision-based navigation algorithms are highly affected from non-nominal conditions, which comprise illumination conditions and environmental uncertainties. Thanks to the outstanding generalization capability and flexibility, deep neural networks (and AI algorithms in general) are excellent candidates to solve the aforementioned shortcoming of navigation algorithms. The paper presents a vision-based navigation system using a Convolutional Neural Network to solve the task of pinpoint landing on the Moon using absolute navigation, namely with respect to the Mean Earth/Polar Axis reference frame. The Moon landing scenario consists in the spacecraft descent on the South Pole from a parking orbit to the powered descent phase. The architecture features an Object Detection Convolutional Neural Network (ODN) trained with supervised learning approach. The CNN is used to extract features of the observed craters that are then processed by standard image processing algorithms in order to provide pseudo-measurements that can be used by navigation filter. The craters are matched against a database con-

taining the inertial location of the known craters. An Extended Kalman Filter with time-delayed measurements integration is developed to fuse optical and altimeter information.
Keywords: Vision-based navigation, Absolute navigation, CNN Craters detection, Lunar landing

1. Introduction

Lunar environment and Artificial Intelligence are becoming increasingly attractive to the Space research community, due to the latest long-term plans of Space Agencies. On one hand, the activities linked to the Lunar Gateway have renovated the deep interest in the mentioned environment [1, 2, 3, 4]. On the other hand, recent advancement in research demonstrate the use of Artificial Intelligence for different tasks in the space domain, from navigation [5, 6] to formation flying guidance and control [7, 8, 9]. In this work, the development of a vision-based navigation system using AI to solve the task of pinpoint landing on the Moon is presented. The considered lunar landing scenario is defined by the spacecraft descent on the South Pole, covering the altitude range from 100 km to 3 km. The proposed navigation strategy is subdivided in two main phases which are slightly overlapped for safety reason: one where absolute navigation is performed, the other estimating instead the relative state with respect to the targeted landing site. The current paper focuses on the absolute navigation scenario. The basic idea is to exploit a coupled architecture, between a navigation filter and an AI-assisted Image Processing (AI-IP) system comprised by neural networks and standard image processing algorithms. The AI-IP will be able of processing images acquired by a navigation camera, generating pseudo-measurements that can be exploited by a sensor-fusion filter, to retrieve the estimated state. In particular, literature demonstrates that craters database matching is typically exploited to retrieve the absolute pseudo-measurement [10, 11]. In these papers, the database matching is performed via nearest neighbor approach, linking the output from the object detector to the navigation filter.

In recent years, the evolution of specialized computing processors, like Tensor Processing Units (TPU) and Vision Processing Units (VPU), and the correspondent increment of available computational power paved the way for future AI-assisted space systems. Some early applications were already tested in orbit [12, 13]. Guidance, Navigation, and Control, and Vision-based navigation in particular, could potentially achieve large benefits from the introduction of such systems.

Different solutions were proposed. In [14] a Convolutional Neural Network (CNN) is coupled with a Long-Short Term Memory (LSTM) to achieve an end-to-end learning for estimating the 6-DoF pose of a UAV during landing. The global position and orientation of the robot are the final outputs of the AI architecture using images and IMU measurements as inputs. A similar concept structure (CNN+LSTM) is proposed and extended by [15], where the final output of the AI system is not the pose estimation, but a thrust control profile to drive the spacecraft in a lunar landing maneuver, mapping the input of the navigation sensors directly to the control action. A Recurrent CNN (RCNN) is adopted in [16] to perform end-to-end 6-DoF visual odometry: the proposed approach exploits a deep learning system based on a monocular visual odometry (VO) algorithm to estimate poses from raw RGB images.

[17] uses a CNN to estimate the depth field of the scene. Information output by the AI is then passed to an optimization algorithm that filters and refine the pose estimation. Craters identification has been proposed as a viable method to perform absolute navigation during lunar landing. State of the art detectors make use of Object Detection Networks (ODN) [18, 19] image segmentation [20, 21, 22]. In order to perform end-to-end navigation, craters detected by the AI requires to be matched with a database. Geometrical [23, 24] or feature-based [25] algorithms were proposed. Finally, object detection algorithms using Deep Convolutional Networks have been proposed for UAV obstacle avoidance in [26].

The objective of the paper is to present the baseline architecture for such intelligent vision-based navigation strategy, together with some preliminary results of the algorithm implementation. In particular, the paper fulfills the following objectives:

- the development of a synthetic image generation pipeline to generate the training dataset and the testing trajectory frames;
- the creation of a customized CNN to work out the task of crater detection in Moon images;
- the derivation of a customized and efficient routine to perform database filtering and matching to retrieve the absolute location of detected craters;
- the development of an Extended Kalman Filter to carry out the navigation estimation based on AI-IP pseudo measurements and altimeter reading, taking into account the measurements delay.

The following sections are structured as follows. In Section 2, the reference Moon landing scenario is presented, detailing the considered landing phases. In Section 3, the proposed complete architecture for the absolute navigation is detailed, from the adopted neural network down to the navigation filter. Simulations results and achieved performance are expounded in Section 4, while conclusions are drawn in Section 5, with highlights for planned future steps.

2. Moon Landing: Scenario Definition

The considered mission scenario consists in the spacecraft descent from an altitude of 100 km down to 3 km targeting the Lunar South Pole area, the designated candidate target for human missions incoming in the next years [27, 28]. Even if the problem of autonomous guidance generation is out of the scope of the present study, feasible trajectories are required to test the proposed AI-based navigation on representative cases. Thus, spacecraft trajectories will be generated executing optimal guidance algorithm depending on the target location and thrust constraints. Moreover, due to the given landing location, it is critical to take into account illumination and shadowing condition. Since the angle between the Moon rotation axis and the ecliptic is close to 90°, in the Polar Regions the topography plays a crucial role in the determination of the illumination conditions. In fact, areas at relatively high altitude can experience continuous periods of illumination (of several months), whereas some crater bottoms are always in shadow. In such scenario, the navigation system can encounter highly varying illumination conditions, with low Sun elevation angle in the South Pole region and large

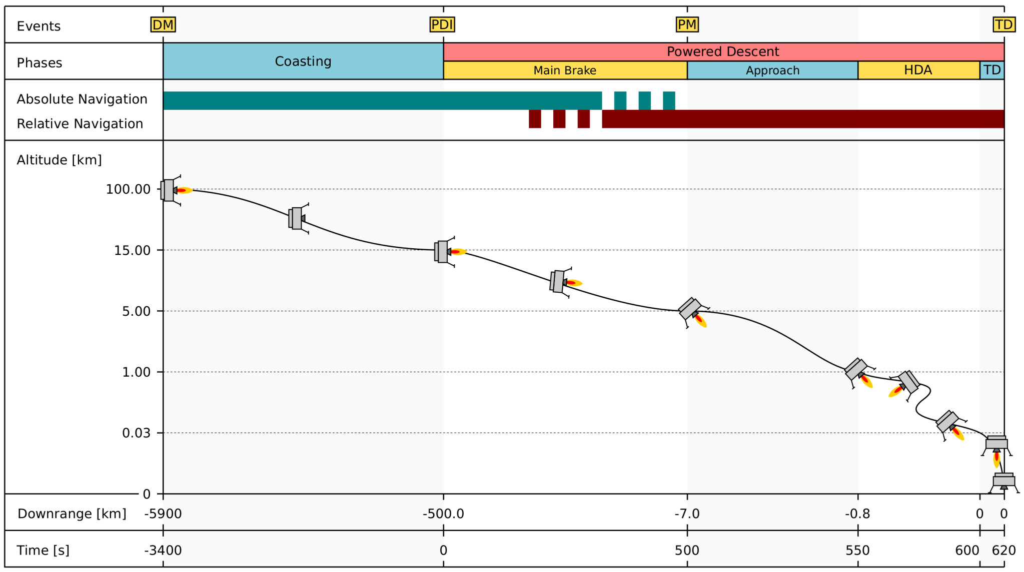


Figure 1: Nominal Lunar Landing Phases and Navigation modes. Distances and times are not in scale. The time scale takes the PDI as origin, while Downrange is assumed to be 0 at the Landing Site.

shadow areas in the image. Figure 1 shows the assumed nominal phases for a Lunar landing mission. It can be seen how on-board Navigation operates in two modes, Absolute and Relative.

The spacecraft is assumed to initiate the landing maneuver from a circular Parking Orbit (PO) with altitude between 100 and 250 km. The spacecraft performs a tangential burn to lower the orbit perilune, inserting itself into an elliptical orbit. The lower the perilune, the lower the overall amount of fuel required for the landing maneuver. At the same time, the terrain topography poses a safety requirement on the minimum altitude of the perilune. 15 km is a generally accepted value and is adopted as nominal value in this study. At the perilune of the transfer orbit the Powered Descent Initiation takes place:

the main thrusters are turned on and the spacecraft performs the Main Brake maneuver, in which most of the horizontal is dropped. The thrust magnitude is constant and close to the maximum. The thrust vector pointing profile is optimized and remains close to the local horizon. During most of this phase the navigation is absolute, while in the last part relative navigation is initialized, for it is required to be already active and running at the beginning of the next phase. As the nominal landing site comes into the field of view of the navigation system, the Final Approach phase begins. The constant thrust constraint is released and the S/C performs a pitch maneuver to point the thrust vector mainly toward ground. In this phase relative navigation is performed; the landing area is scanned and large diversions to the landing trajectory can be commanded to cope with errors. Below 1500 m of altitude, fine trajectory corrections (in the maximum order of magnitude of hundreds of meters) can be ordered to perform the Hazard Avoidance task. This phase ends on the vertical of the selected landing site at a certain altitude (tens of meters), with null horizontal velocity. Pinpoint landing terminates with a powered vertical descent at constant velocity until the touchdown. Absolute navigation, which constitutes the main focus of this work, is operative from the parking orbit until the beginning of the Approach phase.

To ease the formulation of the reference landing maneuver, without loss of generality, in this work is assumed that the inclination of the initial PO can be tuned to match the latitude of the target landing site achieving a planar trajectory (in the inertial reference frame). Nevertheless, the planar assumption in not valid in a Lunar Fixed Frame (LFF), due to the Moon rotation. Such reference frame corresponds to the Mean Earth/Polar Axis

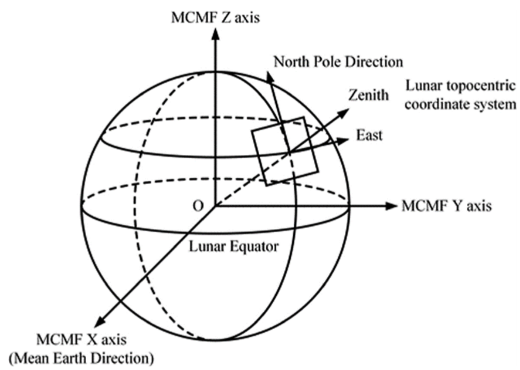


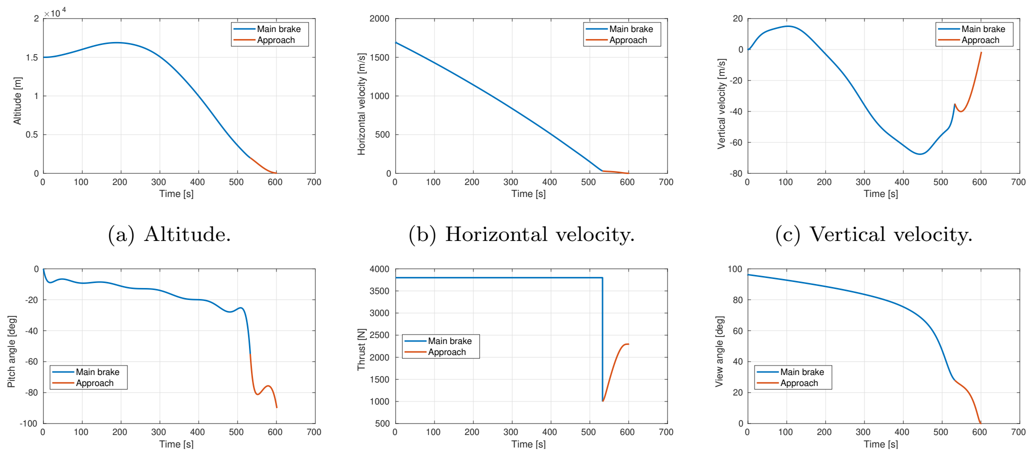
Figure 2: Lunar Fixed Frame Mean Earth/Polar Axis. Figure from [29] by GUO et al.

(ME), as shown in Fig. 2, described in details in [29]. It defines the z-axis as the mean rotational pole, while the Prime Meridian (0° Longitude) is defined by the mean Earth direction. The intersection of the lunar Equator and Prime Meridian occurs at what can be called the Moon's "mean sub-Earth point", due to the Moon's tidal locking to the Earth. In absolute navigation, the state of the spacecraft is reconstructed with respect to the LFF. The effect of the Moon's rotation corresponds to a maximum velocity in the Crossrange direction approximately equal to $4.5\,\mathrm{m\,s^{-1}}$ during the coasting phase. In the models, simulations, and generation of the image dataset used to train, test, and verify the AI system, this effect shall be taken into account.

The times of the transitions between the Main Brake and the Approach, and between the Approach and the Hazard Avoidance, come out from a complex optimization performed in the mission design phase, and depend on the combination of several parameters, related to the controllability of the lander (thrust and torque maximum magnitude, divert capabilities), and to the constraints imposed by hazard detection and navigation capabilities.

2.1. Reference landing maneuver

Figure 3 reports the specific application case used as reference in this work. The lander mass at PDI is assumed to be 1500 kg. The available thrust during the Main Brake is assumed to be 3800 N, while during the Final Approach and in the subsequent phases it is considered to be throttleable between 1000 and 2300 N. The main engine is assumed to be tightly connected to the S/C structure with no thrust vector control, linking the thrust pointing direction directly with the S/C attitude. Trajectory has been optimised to minimize the fuel consumption; a direct optimization method has been used for the main brake phase, combined with the semi-analytical DA guidance described in [30] for the Final Approach. The optimization does not include the coasting phase, which is purely ballistic and in first approximation can be assumed to be a perfect Keplerian arc of an elliptical orbit. Figure 3a shows the nominal altitude profile, starting at the PDI. It can be seen how at the beginning the high tangential speed tends to follow the transfer orbit trajectory and the altitude increases. Then, as the drop in the horizontal velocity becomes relevant, the altitude begins to decrease. Figures 3b and 3c show respectively the horizontal and vertical velocity profiles: most of the horizontal velocity is dropped in the Main Brake, with a velocity in the order of magnitude of $100\,\mathrm{m\,s^{-1}}$ at the beginning of the Approach. The profile of the thrust angle (corresponding to the pitch angle) is shown in Fig. 3d. It is considered to be 0 whenever pointing toward the horizon, and $-90°$ with the thrust vector pointing downward (vertical attitude). The pitch maneuver at the end of the main brake, corresponding to an altitude of $\sim$4000 m is clearly visible. Finally, Fig. 3f reports the view angle on the



(a) Altitude. (b) Horizontal velocity. (c) Vertical velocity.

(d) Thrust angle. (e) Thrust magnitude. (f) View angle over landing site.

Figure 3: Nominal landing maneuver (powered descent and final approach only).

nominal landing site from the lander, an information of particular relevance in a relative navigation scenario, not covered in this work. No specific epoch is assumed for the initiation of the maneuver, nor RAAN for the parking orbit. In this way, the reference maneuver can be adapted to different initial conditions, exposing the navigation system to completely different portions of lunar terrain, giving the possibility to test the system in different conditions.

2.2. Camera and illumination conditions assumptions

A camera with 40° Field Of View (FOV) and a 1024×1024 pixels sensor is assumed as main navigation sensor. During the powered descent the translation of the S/C is controlled in open-loop mode, with the lander tracking a profile of attitude and thrust magnitude computed by the on-board guidance module. In order to cope with error propagation, the trajectory is periodically recalculated, converging progressively to the target. An estimation of the lander state in terms of position and velocity relative to the target is then required only for periodically trajectory update. From past studies, it is known that such system can be effective with a minimum frequency of the trajectory update of 0.2 Hz [30, 31]. Beyond certain frequencies the gain in performance due to the faster update becomes irrelevant. Taking some margin, a minimum requested frequency of 1 Hz is assumed for the image feeding to the navigation.

During the coasting phase, the S/C travels half of the transfer orbit, covering 180 degrees in true anomaly. That implies that the illumination conditions on ground, especially the inclination of the Sun over the terrain, that the navigation system is expected to encounter are extremely variable, from the Sun slightly above the horizon in polar regions to 90° of Sun elevation close to the Lunar Equator. No particular constraints that bounds the Sun inclination to a specific range is assumed.

3. Absolute Navigation Architecture

The absolute navigation task requires the determination of the complete state, i.e. position and velocity with respect to the inertial system fixed to the Moon. The optical measurements are fed to a convolutional neural network, which is trained to identify the database craters present in the image. Such correspondence is later fed to a navigation filter that performs sensor fusion with an altimeter present on-board. A schematic of the architecture is reported in Fig. 4.

3.1. Crater database and AI training set generation

The task of absolute navigation is to estimate the lander position and velocity with respect to the LFF. Thus, the ultimate goal is to retrieve such

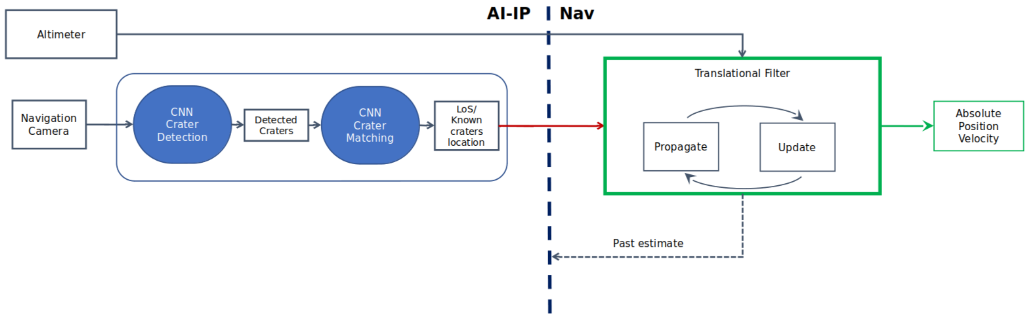


Figure 4: Schematic of absolute navigation modules.

information from the craters identified in the image.

3.1.1. Dataset elements

The objective of the AI algorithm is to detect craters on the lunar surface. For training purposes, a dataset has been created, formed by the following fundamental elements:

- a set of images of the lunar terrain;
- the list of the visible craters in each image, with centers positions and radii in camera coordinates;
- for each image, the correspondent absolute position and attitude of the lander relative to LFF;
- a crater database including the position (or alternatively the latitude/longitude pair) of each labeled crater in the LFF.

3.1.2. Dataset generation framework

An artificial Digital Elevation Model (DEM) of the Moon has been exploited to generate simulated images taken by a landing navigation camera, to create both the AI training dataset and end-to-end simulations of complete landing maneuvers used to evaluate the overall system performances in Section 4. The simulated portion of the Moon used for generating the dataset is a DEM modeled in Pangu, an high fidelity rendering software meant for space applications and realistic rendering of natural celestial bodies [32]. A flat DEM is created, perturbed with fractal noise and enriched with the other relevant terrain features, i.e. craters. A detailed crater database is needed to perform lunar absolute navigation with the proposed architecture. This method allows to have a completely reliable ground-truth for the craters position and size, while real DEMs present craters that are not registered in databases and would not allow to build a completely reliable training set.

In order to generate a rich and representative dataset, the environmental variables in Table 1 are randomly varied within the reported ranges. By doing so, the dataset can cover the wide feature-space that is expected in the operational scenario. The lunar impact crater size and distribution has been extracted from [33].

The craters reported in Table 1 refer to the synthetic generation performed in Pangu. In the ballistic trajectory during the coasting phase, the S/C travels half of the transfer orbit, covering 180° in true anomaly. That implies that the illumination conditions on ground, especially the inclination of the Sun over the terrain, that the navigation system is expected to encounter are extremely variable, from the Sun slightly above the horizon in polar regions to potentially straight illumination with 90° of Sun elevation close to the Lunar Equator. Regarding the Sun Azimuth angle, considering the South Pole region, it is related to the Moon rotation. Therefore, it can

Table 1: Environmental variables and their range of variation for the dataset generation.

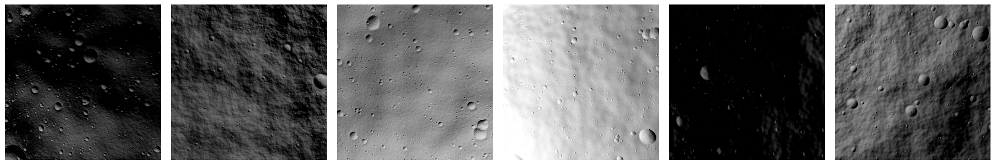
Variable	**Range**
Altitude	3 - 100 km
Attitude pitch (wrt vertical)	$0°$ - $20°$
Sun illumination angle - Elevation	$0°$ - $90°$
Sun illumination angle - Azimuth	$0°$ - $360°$
Synthetic crater frequency	$1.8e6$ - $3e6$
Synthetic craters dimension	6 - 500 m

vary in the whole range between $0°$ to $360°$. Actually, this wide range is applicable only for high latitudes close to the Poles. No particular constraints that bound the Sun inclination to a specific range (like the execution of the whole landing maneuver close to the lunar terminator) are assumed.

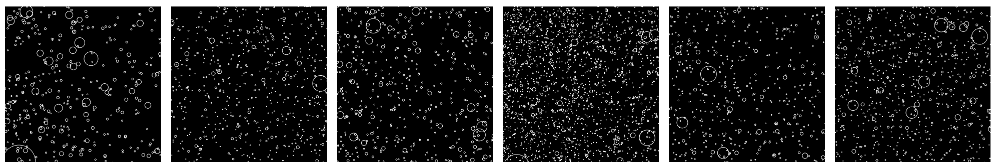
For the dataset, 5000 images have been generated: some examples of images stored in the dataset are reported in Fig. 5a, with the ground-truth craters present in the images highlighted in Fig. 5b.

3.2. Neural network training set

In order to perform the training of the neural network, the dataset described in Section 3.1.2 is completed with the information related to the craters actually visible in each frame. Each image is associated with the list of the craters comprised in the current field of view; the coordinates of each crater are converted into a bounding box, expressed by a vector of coordinates in the image reference frame $[y_{\min}, x_{\min}, y_{\max}, x_{\max}]$. The pairs (image, list of bounding boxes) constitute the actual training set for the subsequent



(a) Original images.



(b) Ground truth craters.

Figure 5: Training set examples.

neural network: the image is the network input, while the list of bounding boxes constitutes the network target.

3.3. CNN for crater detection

Convolutional Neural Networks are particularly suited for replacing effectively some of the traditional image processing algorithms. Several, different network structures can be conceived basing on CNNs. State of the art crater detection networks rely heavily on the so called U-Net architecture [34, 35] to identify landmarks by means of image segmentation [20, 21, 22]. For the first half of its layers, the network downsamples, and then upsamples for the second half, while maintaining short-cut connections between the layers. Finally, it uses template matching to extract the craters from the target masks. Despite its high accuracy, such architecture involves a huge number of trainable parameters, with a computational cost accordingly high. Preliminary tests performed with such architecture confirmed this trend, with a computational burden potentially too high to achieve the target of at least

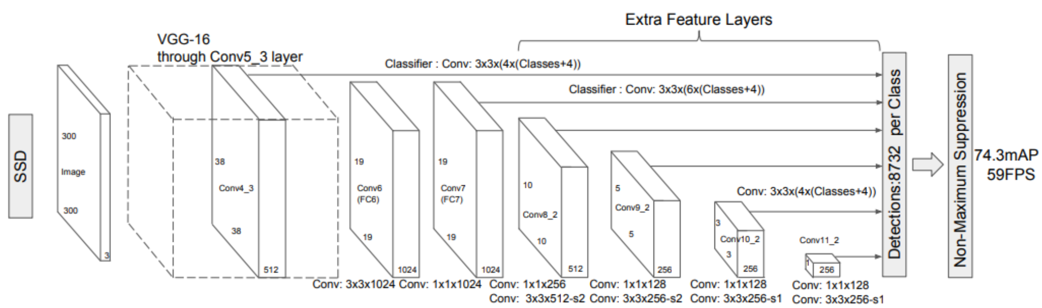


Figure 6: Single Shot Detection (SDD) network from [36] by Liu et al. The original feature extractor is based on VGG-16; a fully convolutional network is than used to estimate both the class and the bounding box associated to each object in the image.

one image processed per second on flight-representative hardware.

To reduce both the computational cost and the training time, a state-of-art Object Detector Network (ODN) is considered. Pre-trained models are already available in Tensorflow Object Detection API framework. The collection has models trained on benchmark datasets such as COCO, KITTI and Open Images: such models are available for initialization of new custom models and to train them on novel datasets. The structure of the network used in the implementation is based on a Single Shot Detector (SDD) [36] with MobileNetV2 [37], pre-trained on the COCO dataset [38], as feature extractor. SDD combined with MobileNetV2 ensures computational efficiency and fast inferences on the embedded system thanks to the use of optimized operations like depth-wise separable convolutions [39] instead of basic convolutions.

The basic structure of a SDD network is shown in Fig. 6: in this work, with respect to the original version represented in the graph, the VGG-16 feature extractor has been replaced with the more computationally efficient

MobileNetV2. Transfer learning allows to exploit the efficient set of low level features that pre-trained networks offer: nevertheless, the specific, selected network imposes some constraints on the size of the input frame, that has to fit the pre-trained architecture. An additional preprocessing step is then requested: first, the image is scaled down to the $320 \times 320\,\mathrm{px}$ resolution, then is converted back in a 3-channel image, achieving a final frame size equal to $320 \times 320 \times 3$, which is the original input size of the MobileNetV2. The conversion from single channel to RGB comes with null computational cost, for the information of the original channel is just replicated 3 times. The downscaling is selected for mainly two reasons: on one hand, in this way, the size of the input is kept unaltered to meet the original input of the MobileNetV2; on the other hand, increasing the input size affects the inference time, which is potentially crucial for the next implementation of the algorithm into real hardware.

The output consists in a $n \times 5$ matrix reporting the list of the craters located in the image. Each row, corresponding to a single detection, is a 5-element vector $[y_{\min}, x_{\min}, y_{\max}, x_{\max}, \alpha]$, where the first four elements consist in the coordinates of the bounding box enclosing the crater. The origin of the reference system is conventionally placed in the upper left corner of the image; values are normalized by image width and height to constrain the interval between zero and one. The index $\alpha \in [0, 1]$ is a score representing the network confidence in the crater identification: low score means little confidence in saying that the output coordinates correspond to the bounding box of a crater. Only craters detected with high confidence $\alpha \geq \overline{\alpha}$ are considered for the subsequent navigation step. The threshold $\overline{\alpha}$ is an adjustable

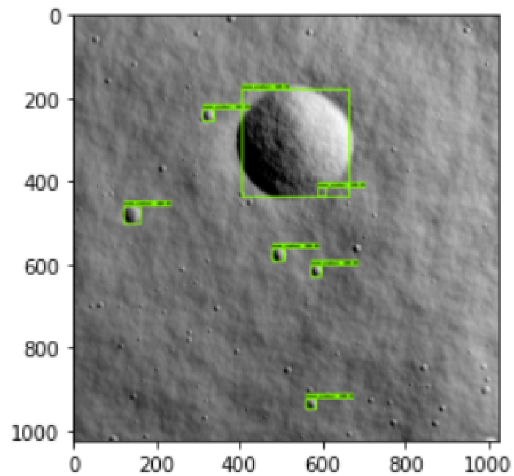


Figure 7: Output of the ODN. Bounding boxes around detected craters.

hyperparameter of the navigation system: in the remainder of the paper, $\overline{\alpha} = 0.6$ is assumed. Standard ODNs are trained to recognize multiple types of objects, and each bounding box has an associated class; in this particular case, the network is trained to detect a single "crater" class. An example of the network output is shown in Fig. 7.

Post-processing is performed on the network output to retrieve the craters coordinates given the bounding boxes predicted by the AI model. The step-wise procedure is summarized here:

- Selection of the predicted bounding boxes according to the confidence threshold $\overline{\alpha}$ and Intersection-Over-Union threshold, $\mathrm{IoU} \geq \overline{\mathrm{IoU}}$ (set to 0.1). Intersection-Over-Union is used in non max suppression, which is used to eliminate multiple boxes that surround the same object, based on which box has a higher confidence. The building blocks of this process are:

- Compare the most confident bounding box with its IoU with every other predicted bounding box of the crater class. If the IoU $\geq \overline{\text{IoU}}$, discard it as it represents a duplicate detection.

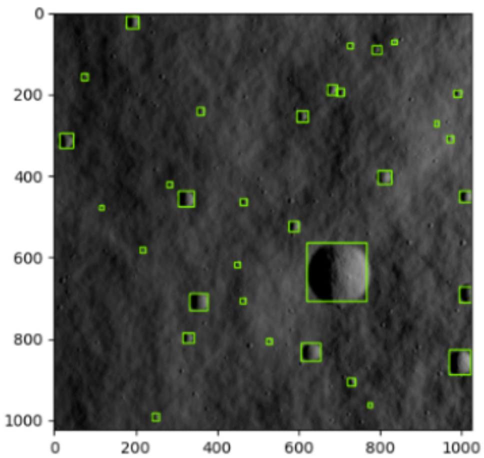
- Remove the output predicted bounding box from the list of bounding boxes.

• Calculate the center and diameter to inscribe each rectangular bounding box into a circle.

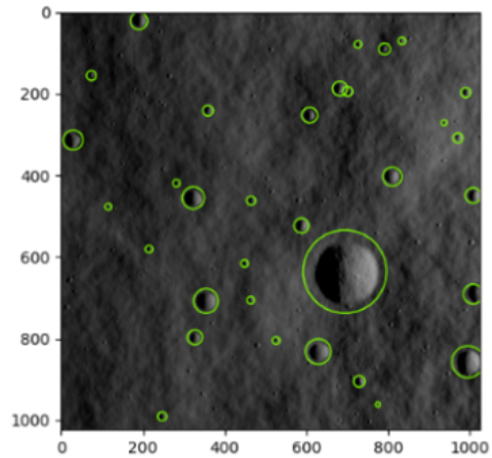
In this way, bounding box coordinates are converted in a 3-element vector $[x, y, \rho]$, where x and y are the coordinates of the crater's center, and ρ its radius, directly comparable with the craters database. An example is shown in Fig. 8. One of the major concerns with CNN and AI in general is their generalization ability, and the proper behavior of the model when performing inferences on real data is not assured when the training is based on synthetic data. Although this aspect needs to be tackled at a systematic level, which is beyond of the scope of this paper, the trained ODN detector has been qualitatively assessed with real Moon images. The results are shown in Fig. 9.

3.4. Database Filtering and Matching

The first task of the navigation module consists in matching the detected craters c_{odn}, expressed in image frame coordinates, to real database craters. In this way, the absolute location of identified features, i.e. craters, can be retrieved. The task requires the feedback knowledge coming from the navigation estimate of the spacecraft position vector. The FOV of the camera

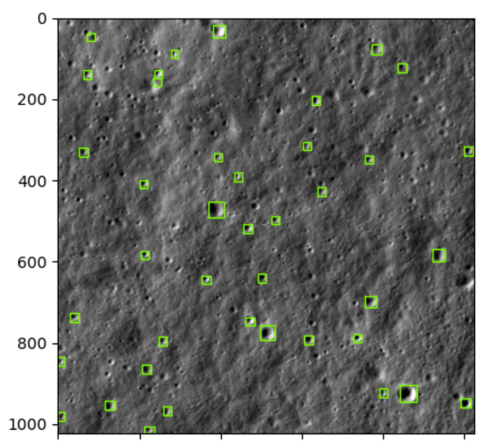


(a) Before the post-processing.

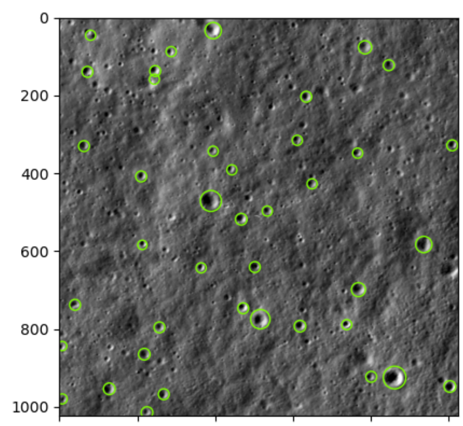


(b) After the post-processing.

Figure 8: ODN post-processing. Bounding boxes are converted into craters' positions and radii.



(a) Real Moon image before the post-processing.



(b) Real Moon image after the post-processing.

Figure 9: Real Moon image ODN post-processing. Bounding boxes are converted into craters' positions and radii. Moon image taken from LROC database [40].

is projected on the surface using spherical projection. In other words, finding the intersection between a sphere and a line one needs to combine the equations:

$$\|\mathbf{x} - \mathbf{c}\|^2 = r^2 \quad (1)$$

where $\mathbf{x}$ are the points on the sphere, $\mathbf{c}$ the center point and r radius of the sphere.

$$\mathbf{y} = \mathbf{o} + d\hat{\mathbf{u}} \quad (2)$$

where $\mathbf{y}$ are points on the line, $\mathbf{o}$ the origin of the line and d the distance from the origin along the line and $\hat{\mathbf{u}}$ the direction of line (a unit vector). The origin is the focal point of the ideal camera.

The output of the projection is a set of corners κ, representing the boundaries of the projected FOV. Each corner is expressed in *latitude* (ϕ) and *longitude* (λ) coordinates. The margined search area $\mathcal{A}$ is a spherical region constructed by extracting the maximum *lat-lon* coordinates out of the κ corners, as shown in Fig. 10. The extracted database craters c_{db} coordinates are first expressed in the inertial frame, then they are projected into the camera frame.

$$\forall c_{db} \in \mathcal{A}, (\lambda, \phi) \rightarrow (X, Y, Z)_I \rightarrow (l_x, l_y, l_z)_C = A_{C/I}(X, Y, Z)_I \quad (3)$$

Finally, the homographic projection is used to retrieve the 2D coordinates of each crater in the image frame, as reported in Eq. 8. At this point, the algorithm possess two lists of craters expressed in the image frame: c_{db} is the query grid out of which the craters in the list c_{odn} are matched.

The matching task is performed using a traditional 1-nearest neighbor routine, the list of craters are organized into a KD-tree to facilitate the

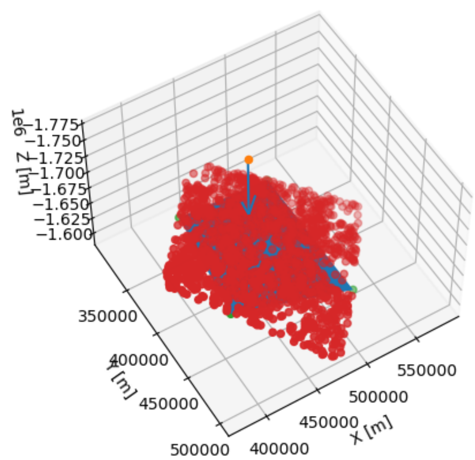


Figure 10: Margined search area $\mathcal{A}$ containing the projected FOV.

matching procedure. The features used for matching are obviously the craters location coupled with the projected diameter. The craters database matching outputs a set of inertial coordinates of craters that have been identified as the groundtruth of the ODN detected craters.

3.5. Absolute Navigation Filter

The absolute navigation task is performed by a discrete-time Extended Kalman Filter, whose output is the estimate of the complete state $\boldsymbol{X}$, composition of position $\hat{\mathbf{R}}$ and velocity $\hat{\mathbf{V}}$ vectors. The inputs received are instead the AI-IP block products and the output of the altimeter, the former sampled at a 1 Hz rate, while the latter at 8 Hz.

The dynamics implemented on the filter, used to retrieve the a-priori estimate of the state, is given by a simplified two-body dynamics expressed in the LFF reference frame with the control component $\boldsymbol{U}$. Eq. 4 reports the complete expression of the right-hand-side of the implemented dynamics $\boldsymbol{f}(\boldsymbol{X})$, where $GM_{☾}$ is the gravitational parameter of the Moon and $\boldsymbol{\Omega} = [0,\ 0,\ \Omega]^{\intercal}$ is the angular velocity of the Moon rotation about its own axis, expressed in the LFF, which coincides with the angular velocity of the non-inertial reference frame.

$$\dot{\boldsymbol{X}} = \boldsymbol{f}(\boldsymbol{X}) = \begin{bmatrix} \dot{R} \\ \dot{V} \end{bmatrix} = \begin{bmatrix} \dot{V} \\ -\frac{GM_{☾}}{\|\boldsymbol{R}\|^3}\boldsymbol{R} + \boldsymbol{V} \wedge \boldsymbol{\Omega} + \boldsymbol{\Omega} \wedge \boldsymbol{R} \wedge \boldsymbol{\Omega} + \boldsymbol{U} \end{bmatrix} \tag{4}$$

The Jacobian matrix deriving from such expression is instead given by Eq. 5.

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} \\ \boldsymbol{F}_{VR} & \boldsymbol{F}_{VV} \end{bmatrix} \tag{5}$$

$$\text{with} \quad \boldsymbol{F}_{VR} = \frac{GM_{☾}}{\|\boldsymbol{R}\|^3} \begin{bmatrix} 3/\|\boldsymbol{R}\|^2 X^2 - 1 & 3/\|\boldsymbol{R}\|^2 XY & 3/\|\boldsymbol{R}\|^2 XZ \\ & 3/\|\boldsymbol{R}\|^2 Y^2 - 1 & 3/\|\boldsymbol{R}\|^2 YZ \\ Sym. & & 3/\|\boldsymbol{R}\|^2 Z^2 - 1+ \end{bmatrix} \tag{6}$$

$$+ \begin{bmatrix} \Omega^2 & 0 & 0 \\ 0 & \Omega^2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\text{and} \quad \boldsymbol{F}_{VV} = \begin{bmatrix} 0 & 2\Omega & 0 \\ -2\Omega & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{7}$$

Concerning instead the measurement models employed, the filter receives from the AI-IP block both the absolute location ρ_i and relative line-of-sight $\mathbf{u}_i$ associated to the i^{th} matched crater. The former is expressed in the LFF frame (see Fig. 2), while the latter is expressed as the two homographic coordinates, projection of the LoS vector onto the on-board camera plane. The measurement function that is implemented on the filter is represented in Eq. 8.

$$\mathbf{u}_i = \begin{bmatrix} u_{i,x} \\ u_{i,y} \end{bmatrix} = f/l_{i,z} \begin{bmatrix} l_{i,x} \\ l_{i,y} \end{bmatrix} \tag{8}$$

$$\text{with} \quad \mathbf{l}_i = \mathbf{A}_{C/I} \frac{\hat{\mathbf{R}} - \rho_i}{\left\| \hat{\mathbf{R}} - \rho_{\mathbf{i}} \right\|} \tag{9}$$

Here some parameters have been introduced, namely the focal length f, the LoS vector in the camera frame $\mathbf{l}_i$ and the rotation matrix $\mathbf{A}_{C/I}$ from the LFF frame (I) to the camera frame (C). Given the necessity of retrieving the absolute attitude of the spacecraft, a rotational filter is required to be running in synchronous advance with respect to the translational filter.

These pieces of information are fused together with measurements of the satellite altitude ζ, taken by an altimeter. As such, the measurement model reported in Eq. 10, where $R_{☾}$ is the average Moon radius.

$$\zeta(\boldsymbol{R}) = \|\boldsymbol{R}\| - R_{☾} \tag{10}$$

The complete measurement function is then constructed as the collection of all the homographic coordinates associated to the detected craters as per Eq. (8) with the addition of the altimeter measure estimate, as in Eq. (10). The resulting measurement estimate vector $\boldsymbol{h}$ is composed by $2N_{crat} + 1$ elements, where N_{crat} is the number of detected craters. The associated Jacobian matrix $\boldsymbol{H}$ will have a size of $(2N_{crat} + 1 \times 6)$ and is to be assembled with N_{crat} (2×6) $\boldsymbol{H}_{i,crat}$ matrices for each crater LoS and with a single

(1×6) $\boldsymbol{H}_{alt}$ for the altimeter part. The complete expressions are omitted due to their cumbersome representations.

Algorithm 1 reports the most relevant steps of the EKF procedure in a pseudo-code format.

Algorithm 1 Extended Kalman Filter

$\hat{\boldsymbol{X}}_k^- = \int_{t_{k-1}}^{t_k} f(\boldsymbol{X}(\tau))d\tau, \quad \boldsymbol{X}_{k-1} = \hat{\boldsymbol{X}}_{k-1}, \quad \hat{\boldsymbol{X}}_0^+ = \boldsymbol{X}_0$ ▷ Absolute state propagation

$\boldsymbol{F}_k = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{X}}\Big|_{\hat{\boldsymbol{X}}_{\boldsymbol{k-1}}}, \quad \boldsymbol{H}_k = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{X}}\Big|_{\hat{\boldsymbol{X}}_{\boldsymbol{k}}^-}$ ▷ State and measurement Jacobian matrices

$\boldsymbol{\Phi}(t_k, t_{k-1}) = \boldsymbol{I}_{6x6} + \boldsymbol{F}_k \Delta t$ ▷ State Transition Matrix

$\boldsymbol{P}_k^- = \boldsymbol{\Phi}(t_k, t_{k-1})\boldsymbol{P}_{k-1}^+ \boldsymbol{\Phi}^T(t_k, t_{k-1}) + \boldsymbol{Q}, \quad \boldsymbol{P}_0^+ = \boldsymbol{P}_0$ ▷ State Covariance matrix propagation

$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{H}_k^T (\boldsymbol{H}_k \boldsymbol{P}_k^- \boldsymbol{H}_k^T + \boldsymbol{R}_k)^{-1}$ ▷ Kalman gain matrix computation

$\hat{\boldsymbol{X}}_k^+ = \hat{\boldsymbol{X}}_k^- + \boldsymbol{K}_k(\boldsymbol{Y}_k - h(\boldsymbol{X}_k^-))$ ▷ Absolute State correction

$\boldsymbol{P}_k^+ = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k)\boldsymbol{P}_k^- (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k)^T + \boldsymbol{K}_k \boldsymbol{R} \boldsymbol{K}_k^T$ ▷ State Covariance matrix correction

The output of the filter is then fed back in the IP block. The estimate of the state can be exploited in order to restrain the research space for the pattern matching algorithms. However this additional information for the IP block could not be exploited in non-nominal conditions, such as a *lost-in space* scenario, reason for which performance assessment without this improvement shall be performed as well.

3.6. Time-Delayed Measurements Fusion

The navigation algorithm heavily relies on optical measurements. The information content is extracted from the images through the ODN and the intermediate post-processing. Such process takes a finite amount of time that needs to be taken into account when fusing the measurements in the Extended Kalman Filter, especially for real-time applications. Indeed, when delayed measurements are presents, at instant k the system receives a delayed measurement corresponding to time instant s ($s = k - N$, where N number of delay samples). In this paper, a known delay of 1 s is reached in light of the future hardware implementation. There are various methods to consider the measurements delays in the navigation filter:

- Filter recalculation method: it consists of coupling two filters running at fast and slow rate [41]. The former incorporates the high-frequency measurements, whereas the latter is activated every time a delayed (e.g. slow and less frequent) measurement arrives. The method computes the entire trajectory of the state until the current step. Using this method, optimality is guaranteed at the cost of computational burden.

- Alexander Method: it consists on updating the covariance matrices at time s as if the delayed measurement arrived. Then, once measurements $\mathbf{Y}_s$ are inserted at time k, the update is simply the standard Kalman Filter one with a correction matrix term [42].

- Larsen Extrapolation Method: The method described in [42] requires the measurement matrix $\mathbf{H}_s$ and the noise distribution matrix $\mathbf{R}_s$ at time s. In the presented scenario, this is not valid: indeed, the measurement matrix depends on the relative positioning of the camera and craters. Larsen developed a measurement extrapolation method that does not require knowledge about the two matrices until time k [43]. Such method is taken as reference to implement a modified version suitable for the analyzed scenario.

The adaptation of the Larsen method for the measurement fusion is hereby described. For details on the derivation, the reader is suggested to refer to the original reference [43]. Several modifications were needed to solve two shortcomings of the original method: the incorporation of high-frequency altimeter and the extension to the nonlinear Extended Kalman Filter. For the former, the filter firstly computes the gain and the updates as in Algorithm 1 fusing fast altimeter measurements. For what concerns the delayed measurements, let us call the measurements coming from the time instant $s = k - N$ as $\mathbf{Y}_s$, which are incorporated at time instant k. The Larsen method consists in calculating an extrapolated measurements from $\mathbf{Y}_s$ to be integrated at time k, called $\mathbf{Y}_{k,s}^{ext}$:

$$\mathbf{Y}_{k,s}^{ext} = \mathbf{Y}_s + h(\hat{\mathbf{X}}_k^-) - h(\hat{\mathbf{X}}_s^+) \tag{11}$$

At each intermediate step between s and k a correction term M is calculated as:

$$\mathbf{M}_k = \left[\prod_{i=0}^{k-s-1} (\mathbf{I} - \mathbf{K}_{k-i}\mathbf{H}_{k-i})\mathbf{\Phi}(t_{k-i}, t_{k-i-1}) \right] \mathbf{P}_s \tag{12}$$

where the Kalman gain and measurement sensitivity matrix $\mathbf{H_{k-i}}$ at step $k-i$ does not reflect any update coming from the delayed measurement $\mathbf{Y}_s$. Then, the updates of the correction term are calculated as follows, modifying the correction equations in Algorithm 1:

$$\mathbf{K}_{k,s} = \mathbf{M}_k \mathbf{H}_{k,s}^T [\mathbf{H}_{k,s}\mathbf{P}_s\mathbf{H}_{k,s}^T + \mathbf{R}_s]^{-1} \tag{13}$$

$$\hat{\mathbf{X}}_k^+ = \hat{\mathbf{X}}_k^- + \mathbf{K}_{k,s}(\mathbf{Y}_{k,s}^{ext} - h(\hat{\mathbf{X}}_k^-)) \tag{14}$$

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_{k,s}\mathbf{H}_{k,s}\mathbf{M}_k^T\mathbf{P}_k^{-1})\mathbf{P}_k^-(\mathbf{I} - \mathbf{K}_{k,s}\mathbf{H}_{k,s}\mathbf{M}_k^T\mathbf{P}_k^{-1})^T + \mathbf{K}_{k,s}\mathbf{R}_s\mathbf{K}_{k,s}^T \tag{15}$$

The covariance update is a modified version of the Joseph formula adapted to the original Larsen method. This is done to ensure that the covariance matrix remains positive semi-definite. As seen in Eq. 11 and Eq. 12, the extrapolation method always requires only two matrix multiplications at each time instant and the storage of two variables any time an image is acquired.

4. Navigation Training and Test

In order to assess and validate the performances of the proposed architecture a testing pipeline has been put in place, exploiting the following blocks.

High-fidelity dynamics Simulator. This block takes as input a nominal guidance profile, like the one described in Section 2.1, and simulates the overall maneuver from lander trajectory to the sensors readings in a high-fidelity scenario. The output consists in the ground-truth trajectory and in simulated measurement histories for attitude and additional navigation sensors (i.e. altimeters) necessary for algorithm validation. Since the whole maneuver takes place in close proximity to the lunar ground, only the Moon gravitational pull is included in the translational dynamics. The LP165P spherical harmonics model up to the 165^{th} order [44] is adopted. Disturbances in both direction and magnitude of thrust are included. The nominal thrust vector is rotated by a random angle with normal distribution with zero mean and standard deviation $\sigma = 1°$. Thrust magnitude is perturbed by a Gaussian noise with standard deviation 23 N (1 % of the assumed throttleable thrust). The proposed navigation system provides an estimate of the translation states only, but relies on attitude determination to identify the camera pointing direction: then, attitude estimation errors could have an impact on navigation performances. The spacecraft rotational dynamics are not simulated: the navigation camera is assumed to maintain a nominal nadir pointing, while a Gaussian noise with standard deviation $\sigma = 1°$ is added on the three Euler Angles to represent attitude determination errors. A zero mean Gaussian white noise is added to the ground-truth altitude to simulate altimeter measures. A standard deviation of 1 % of the current altitude is assumed, reflecting the actual behavior of the laser altimeter technology. The whole model is implemented in a *Matlab-Simulink* environment, with altitude measurements generated at frequency 8 Hz. An example of simulated altitude measurement is reported in Fig. 11, compliant with the scenario described in Section 2.

Navigation images rendering. As reported in section 3.1, a 3D rendering tool is adopted to simulate realistic images as generated by the on-board navigation camera. *Pangu* is exploited for such purpose, taking as input the ground-truth generated by the simulator block for both trajectory and spacecraft attitude and outputting a sequence of images sampled at 1 Hz.

AI-IP block. This block includes the application of the trained crater detector to the generated images to extract the centers and the radii of the detected craters, expressed in pixels coordinates in the camera frame. This procedure is executed in a *Python* environment.

Navigation algorithm simulator. The final block is instead in charge of running the crater matching and the filtering tasks in a step-wise fashion, exploiting as inputs all the generated measurements, i.e. the ODN products and the altimeter readings, dealing also with the two different sampling rates. Each second of simulation, the algorithm given its current best state estimate, runs the crater matching procedure, comparing the detected craters to the pruned craters database and pass to the filter the measurements of all the

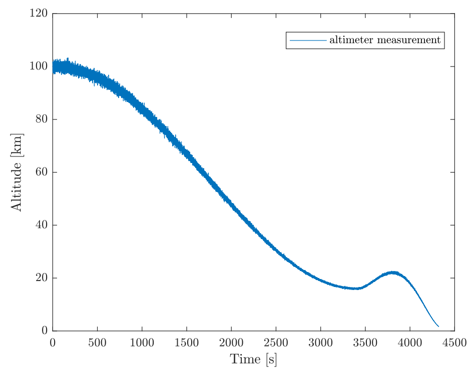


Figure 11: Simulated measured altitude.

matches retrieved. The state estimate is then be compared to the ground-truth trajectory and, together with the estimated covariance matrix, used to assess the overall strategy performances. Also this block is implemented in a *Matlab-Simulink* environment.

4.1. CNN training

The use of a pre-trained MobileNetV2 model allowed to rely on transfer learning to speed-up the training process, achieving at the same time good results even with a relatively small training set like the one described in Section 3.1. A fine-tuning of the original weights of the last two network branches was sufficient for converging to an optimal solution, while all the other weights were maintained fixed at the original value. No data augmentation was applied or required. The adopted loss function replicates the original implementation by Liu et al. [36], and it is composed by two contributions to be minimized concurrently:

$$L(x, c, l, g) = \frac{1}{N}\big(L_{\text{conf}}(x, c) + \alpha L_{\text{loc}}(x, l, g)\big) \tag{16}$$

where L_{loc} is a localization loss that reduces the L1 distance between the predicted box l and the ground-truth box g, and L_{conf} is a confidence loss that expresses the confidence level for a particular image crop to pertain to a particular class c. The x variable is an indicator for matching the ith default box for the jth ground-truth box of category p.

The network is trained on the dataset presented in Section 3.1, with batch size optimized to 16 samples, and learning rate optimized to 0.001, using the Adam optimizer [45]. The train-validation split equals to 80-20%. Test images are those generated for the trajectories: sequential images coming from the same Pangu rendered Moon region. Transfer learning considerably speeds-up the training process, avoiding the burden to learn low-level features, inherited from the pre-trained network. Actual learning, enabled for higher layers only, tailors the network over the specific problem even with a relatively small dataset: the loss function reached its minimum in 150 epochs. Relevant statistical metrics on the test dataset are reported in Tab. 2.

Table 2: ODN statistical performance metrics on the test dataset.

Metric	**Value**
Precision	0.6
Recall	0.9
F1-score	0.7
Mean IoU	0.7

4.2. Craters detection and matching performance

In this section, the assessment of the network capabilities in detecting craters, and of their subsequent matching within the crater database is presented. Fig. 12 shows the results of the crater matching task for a sample image never seen by the network during the training phase. Craters detected by the ODN (red) are matched to the ground truth database (yellow/green). On the test set, the average correct match percentage is nearly $\sim$75 % in each frame. Craters detection delivers an average localization error of $\sim$3 px in the LoS, as reported in Fig. 14, in which the average center location error for each frame associated with the relative standard deviation of the error distribution is shown. Please note that such error is only related with the ODN crater detection on each frame.

The radius fit output by the network is generally larger than the ground-truth, as shown in Fig. 13 for an example image. A possible motivation lies in the ODN post-processing stage, in which the crater radius is computed by assuming the bounding box inscribed into a circular shape. Nevertheless, this issue does not affect the absolute navigation itself, but rather the matching process with the database given that the KD-tree search is performed in the three dimensions $[x, y, \rho]$. Figure 15 shows the histogram of the radius estimation error: the average error is $\sim$15 % with respect to the ground-truth.

The number of detected craters is deemed as sufficient for the required accuracy; indeed, it is in the order of $\sim$ 50 detected craters for each frame. Figures 16 and 17 show the frequency of the number of detected craters in the prototyping trajectory.

An inference time on the network of 0.02 s per image on a Nvidia RTX

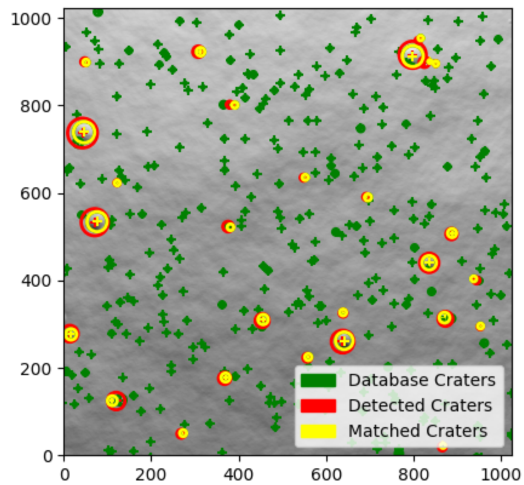


Figure 12: Example of crater detection and matching. The green craters are those extracted from the database. Given the limited size, they are discarded from the matching procedure.

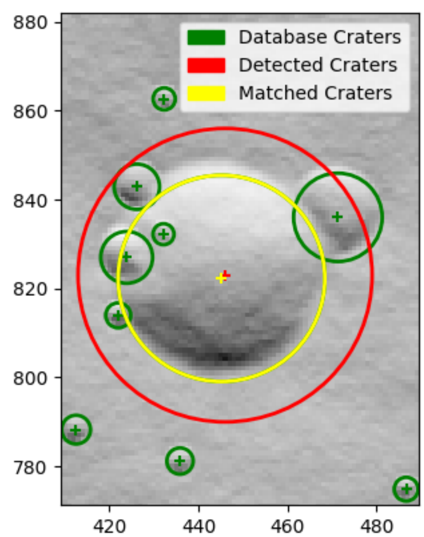


Figure 13: Wrong estimation of crater diameter. The detected diameter is larger than the database ones, due to the ODN post-processing.

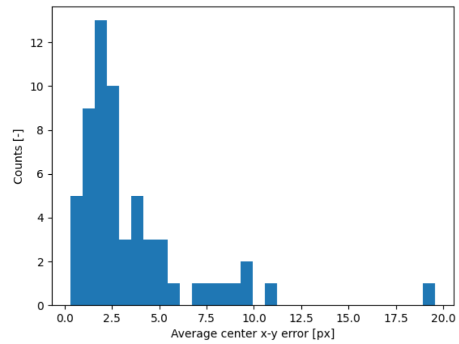


Figure 14: LoS detection errors in pixel between database and detected craters.

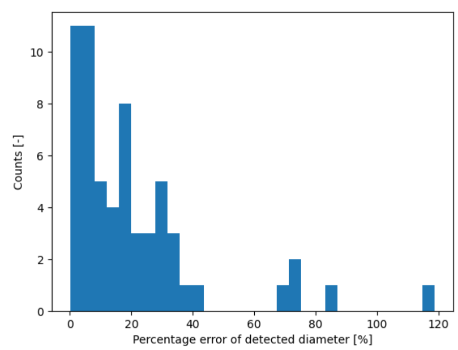


Figure 15: Diameter estimation error in detected craters.

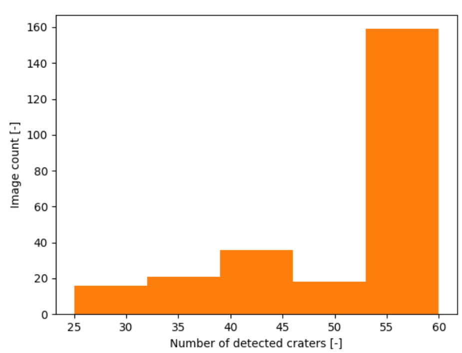


Figure 16: Detected craters distribution along a portion of the prototyping trajectory.

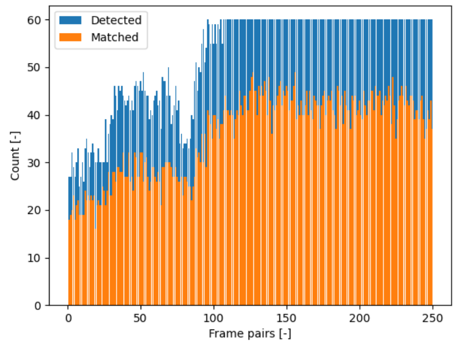


Figure 17: Detected and matched craters distribution along a portion of the prototyping trajectory.

Titan X GPU (or 57 frame per seconds, with FP32 model precision) was preliminary assessed. No model optimization has been performed for the test execution. In order to obtain a better estimation of the potential performance in flight, the network has been implemented also on a Raspberry Pi 4 equipped with an Intel Movidius Neural Compute Stick. This system is built around the Intel Myriad 2 VPU, a possible future architecture for AI systems in space applications, that has already proven effective in flight [12, 13]. On Myriad hardware, with no code optimization, the inference time increases to 0.4 s per image (or 2.5 frames per seconds), a value compatible with the needs of autonomous navigation.

4.3. Navigation Prototyping: numerical results

The full pipeline coupling AI-IP and NAV module has been tested using a sample trajectory, resembling the scenario described in section 2. The craters database (cfr. [46]) linked to the absolute navigation is used down to an altitude of roughly ∼20 km. Below such altitude, the database spatial density becomes lower than the captured FOV on the surface resulting in very few craters to be matched. The issue can be easily solved by refining the database list or coupling the absolute navigation with a relative navigation module that performs frame-to-frame motion estimation. The tuning parameters used for the simulations are reported in Tab. 3.

Figure 18 reports an instant in time, showing the different navigation modules performing the task. The database filtering restrains the search region in longitude and latitude coordinates, then the detected craters are matched using location and diameter as descriptors. Please note that the number of detected and processed craters is limited by the maximum craters variable in Tab. 3, which aims at keeping the number of processed craters within an acceptable range for navigation performance without overburdening the computational cost for subsequent real hardware implementation. The navigation estimate is reported in Fig. 19. The horizontal and vertical error are reported instead of the three-axis results. The vertical error is the projection of the error $\Delta = \mathbf{R}_{nav} - \mathbf{R}_{gt}$ along the radial direction. The norm of the along-track and across-track error is combined in the horizontal one. The navigation yields an estimation error $\sim 200\,\mathrm{m}$ along the trajectory, both for vertical and horizontal error, which is aligned with the expected performance at these relevant altitudes. The small peak present in the plot refers to the passage on the North Pole. Such degradation is due to the fact that, in that area, the database filtering is performed on a spherical cap rather than a margined projected FOV. This is done at high latitude (i.e. above 88°) in order to avoid any singularities or wrapping errors at the polar point.

Parameter	Value	Description
MAX_N_CRAT	50	Maximum number of processed craters
P_0	$\text{diag}(10^4\mathbf{I}_{3\times3}, 10^0\mathbf{I}_{3\times3})$	Initial Covariance Matrix
R_{elem}	$10^4\mathbf{I}_{2\times2}$	Elementary crater localization error covariance
R_{alt}	10^2	Elementary altimeter error variance

Table 3: Simulation parameters

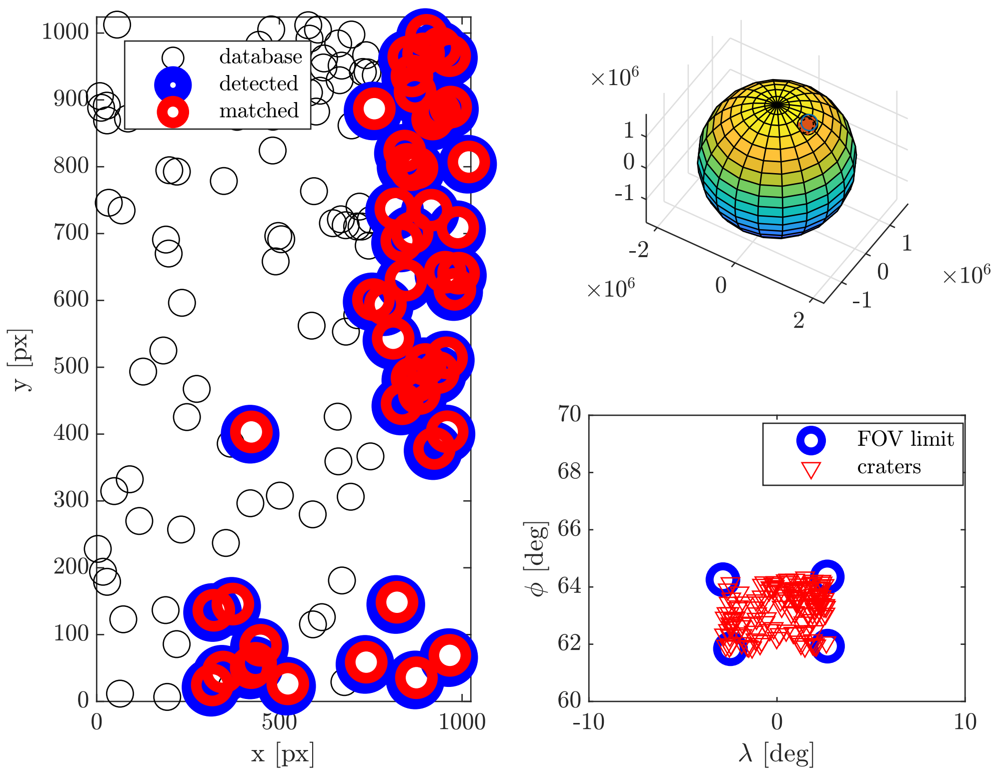


Figure 18: Running navigation performing λ-ϕ database searching and matching. The database filtering restrains the search region in longitude and latitude coordinates (bottom right), then the detected craters are matched using location and diameter as descriptors (top left).

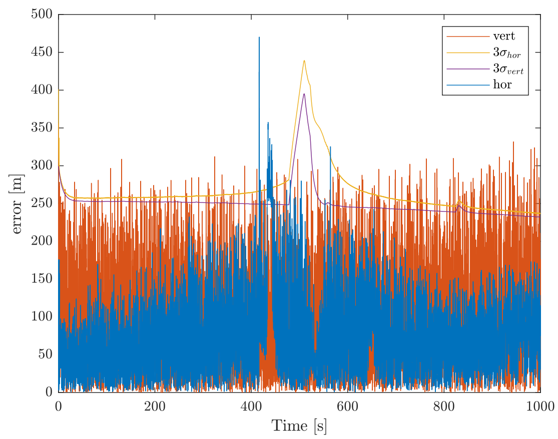


Figure 19: Estimation horizontal and vertical error for absolute navigation.

5. Conclusion

The paper proposed an AI-based Optical Navigation algorithm to perform absolute navigation during Lunar landing. A successful integration of both the detection ODN and the matching and estimation algorithm has been presented. The ODN network can successfully retrieve the Moon craters in an image. The crater detection delivers excellent center localization results (below ~ 3 px) with respect to database ones, on average. This result provides the required performance for the subsequent matching task. The crater diameters are slightly over-estimated due to ODN post-processing, nevertheless the prototyping tests did not show any criticalities for such behavior. However, for future development, an additional consolidation may include a RANSAC-like algorithm to filter out the outliers matching.

The database crater matching for absolute navigation has been developed and tested: results showed that each detected crater was matched to database ones, with a small percentage of false matches. The whole pipeline for absolute navigation, including AI-IP e NAV filter, has been implemented and tested in a sample trajectory at prototype level, taking into account the injection of delayed measurements. The sample scenario demonstrated that the navigation system can meet the performance requirements. Moreover, the complete pipeline for dataset generation has been set-up and used to create the prototyping dataset.

The presented work foresees implementation on real hardware with delicate execution time constraints, hence it is critical to reduce the computational burden on-board. The main objective is to assess the applicability of CNN-based crater detector to the absolute navigation task. In particular, the usage of such ODN architecture, compared with classical algorithms, may be beneficial under several aspects: first of all, the amount of crater detections that ODN-detector produces is robust to disturbances or modifications of the image quality due to Gaussian noise, shot noise, brightness levels, as well as different illumination conditions. This robustness to visual alterations supports the feasibility of CNN-based techniques as reliable navigation architectures that do not imply a high level of human input or tuning. This is confirmed also by other works [21, 22, 47, 16]. Moreover, as mentioned, this work is aimed at the subsequent integration in real flight-like hardware: the achieved inference time makes the ODN a promising, and fast, alternative to the iterative, and slower, processes of template matching and thresholding required in classical techniques [10]. In this way, and end-to-end navigation cycle, including the image processing, can be achieved at 1 Hz frequency.

Acknowledgments

The presented work was funded by the European Space Agency, under the ESA Contract No. 4000129466/19/NL/CRS. The authors would like to acknowledge the support from the AIVIONIC consortium: DEIMOS Engenharia S.A. (Portugal) as prime contractor; AIKO SRL (Italy), Politecnico di Milano (Italy), Fortiss Gmbh (Germany), Ubotica (Ireland) and DEIMOS Space S.L.U. (Spain).

References

[1] A. Colagrossi, M. Lavagna, Dynamical analysis of rendezvous and docking with very large space infrastructures in non-Keplerian orbits, CEAS Space Journal 10 (1) (2018) 87–99. doi:10.1007/s12567-017-0174-4.

[2] A. Colagrossi, V. Pesce, L. Bucci, F. Colombi, M. Lavagna, Guidance, navigation and control for 6DOF rendezvous in Cislunar multi-body environment, Aerospace Science and Technology 114 (2021) 106751. doi:10.1016/j.ast.2021.106751.

[3] F. Colombi, A. Colagrossi, M. Lavagna, Characterisation of 6DOF natural and controlled relative dynamics in cislunar space, Acta Astronautica (2021). doi:10.1016/j.actaastro.2021.01.017.

[4] S. Li, P. Cui, H. Cui, Vision-aided inertial navigation for pinpoint planetary landing, Aerospace Science and Technology 11 (6) (2007) 499–506. doi:https://doi.org/10.1016/j.ast.2007.04.006.

[5] V. Pesce, S. Silvestrini, M. Lavagna, Radial basis function neural network aided adaptive extended Kalman filter for spacecraft relative navigation, Aerospace Science and Technology 96 (2020) 105527. doi:10.1016/j.ast.2019.105527.

[6] A. Pasquale, S. Silvestrini, A. Capannolo, P. Lunghi, M. Lavagna, Small bodies non-uniform gravity field on-board learning through Hopfield Neural Networks, Planetary and Space Science 212 (December 2021) (2022) 105425. doi:10.1016/j.pss.2022.105425.

[7] S. Silvestrini, M. Lavagna, Neural-aided GNC reconfiguration algorithm for distributed space system: development and PIL test, Advances in Space Research 67 (5) (2021) 1490–1505. doi:10.1016/j.asr.2020.12.014.

[8] S. Silvestrini, M. Lavagna, Neural-based predictive control for safe autonomous spacecraft relative maneuvers, Journal of Guidance, Control and Dynamics (2021). doi:10.2514/1.G005481.

[9] S. Silvestrini, M. Lavagna, Relative Trajectories Identification in Distributed Spacecraft Formation Collision-Free Maneuvers using Neural-Reconstructed Dynamics, in: AIAA Scitech 2020 Forum, no. January, 2020, pp. 1–14. doi:10.2514/6.2020-1918.

[10] L. Singh, S. Lim, On lunar on-orbit vision-based navigation: Terrain mapping, feature tracking driven EKF, AIAA Guidance, Navigation and Control Conference and Exhibit (August) (2008) 1–18. doi:10.2514/6.2008-6834.

[11] C. Xu, X. Huang, M. Li, D. Wang, Landmark database selection for vision-aided inertial navigation in planetary landing missions, Aerospace Science and Technology 118 (2021) 107040. doi:https://doi.org/10.1016/j.ast.2021.107040. URL https://www.sciencedirect.com/science/article/pii/S1270963821005502

[12] G. Giuffrida, L. Diana, F. de Gioia, G. Benelli, G. Meoni, M. Donati, L. Fanucci, Cloudscout: A deep neural network for on-board cloud detection on hyperspectral images, Remote Sensing 12 (14) (2020). doi:10.3390/rs12142205.

[13] G. Mateo-Garcia, J. Veitch-Michaelis, L. Smith, S. V. Oprea, G. Schumann, Y. Gal, A. G. Baydin, D. Backes, Towards global flood mapping onboard low cost satellites with machine learning, Scientific Reports 11 (2021) 7249. doi:10.1038/s41598-021-86650-z.

[14] F. Baldini, A. Anandkumar, R. M. Murray, Learning pose estimation for uav autonomous navigation and landing using visual-inertial sensor data, in: 2020 American Control Conference (ACC), 2020, pp. 2961–2966. doi:10.23919/ACC45564.2020.9147400.

[15] R. Furfaro, I. Bloise, M. Orlandelli, P. Di Lizia, F. Topputo, R. Linares, Deep learning for autonomous lunar landing, in: 2018 AAS/AIAA Astrodynamics Specialist Conference, Snowbird, UT, 2018.

[16] S. Wang, R. Clark, H. Wen, N. Trigoni, Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks,

in: 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 2043–2050. doi:10.1109/ICRA.2017.7989236.

[17] E. Brachmann, C. Rother, Learning less is more - 6d camera localization via 3d surface regression, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 4654–4662. doi:10.1109/CVPR.2018.00489.

[18] E. Emami, T. Ahmad, G. Bebis, A. Nefian, T. Fong, Lunar crater detection via region-based convolutional neural networks, in: 49th Lunar and Planetary Science Conference, The Woodlands, TX, 2018.

[19] E. Emami, T. Ahmad, G. Bebis, A. Nefian, T. Fong, Crater detection using unsupervised algorithms and convolutional neural networks, IEEE Transactions on Geoscience and Remote Sensing 57 (8) (2019) 5373–5383. doi:10.1109/TGRS.2019.2899122.

[20] A. Silburt, M. Ali-Dib, C. Zhu, A. Jackson, D. Valencia, Y. Kissin, D. Tamayo, K. Menou, Lunar crater identification via deep learning, Icarus 317 (2019) 27–38. doi:10.1016/j.icarus.2018.06.022.

[21] L. Downes, T. J. Steiner, J. P. How, Deep Learning Crater Detection for Lunar Terrain Relative Navigation, in: AIAA Scitech 2020 Forum, 2020, pp. 1–12. doi:10.2514/6.2020-1838.

[22] L. M. Downes, T. J. Steiner, J. P. How, Lunar terrain relative navigation using a convolutional neural network for visual crater detection, in: 2020 American Control Conference (ACC), 2020, pp. 4448–4453. doi:10.23919/ACC45564.2020.9147595.

[23] W. Park, Y. Jung, H. Bang, J. Ahn, Robust crater triangle matching algorithm for planetary landing navigation, Journal of Guidance, Control, and Dynamics 42 (2) (2019) 402–410. doi:10.2514/1.G003400.

[24] J. qing Chen, P. yuan Cui, H. tao Cui, Automated detection and classification for craters based on geometric matching, in: J. C. Zarnecki, C. A. Nardell, R. Shu, J. Yang, Y. Zhang (Eds.), International Symposium on Photoelectronic Detection and Imaging 2011: Space Exploration Technologies and Applications, Vol. 8196, International Society for Optics and Photonics, SPIE, 2011, pp. 543 – 548. doi:10.1117/12.901020.

[25] W. Shao, J. Xie, L. Cao, J. Leng, B. Wang, Crater matching algorithm based on feature descriptor, Advances in Space Research 65 (1) (2020) 616–629. doi:10.1016/j.asr.2019.09.031.

[26] R. Opromolla, G. Fasano, Visual-based obstacle detection and tracking, and conflict detection for small UAS sense and avoid, Aerospace Science and Technology (2021) 107167doi:https://doi.org/10.1016/j.ast.2021.107167.

[27] NASA's lunar exploration program overview, retrieved August 7, 2021 (2020).
URL `https://www.nasa.gov/sites/default/files/atoms/files/artemis\_plan-20200921.pdf`

[28] P. Tripathi, R. Garg, Spectral characterization of shoemaker crater at the lunar south pole: possible science objective for the artemis iii human

mission to moon, in: ACRS 2020 - 41st Asian Conference on Remote Sensing, 2020.

[29] H. Guo, W. Fu, G. Liu, Frontiers of moon-based earth observation, in: Scientific Satellite and Moon-Based Earth Observation for Global Change, Springer, 2019, pp. 541–590.

[30] P. Lunghi, R. Armellin, P. Di Lizia, M. Lavagna, et al., Semi-analytical adaptive guidance computation based on differential algebra for autonomous planetary landing, in: 26th AAS/AIAA Space Flight Mechanics Meeting, Vol. 158, Univelt, 2016, pp. 2003–2022.

[31] P. Lunghi, M. Lavagna, R. Armellin, A semi-analytical guidance algorithm for autonomous landing, Advances in Space Research 55 (11) (2015) 2719–2738. doi:10.1016/j.asr.2015.02.022.

[32] I. Martin, M. Dunstan, M. Sanchez Gestido, Planetary surface image generation for testing future space missions with pangu, in: 2nd RPI Space Imaging Workshop, Sensing, Estimation, and Automation Laboratory, United States, 2019.

[33] G. Neukum, B. König, J. Arkani-Hamed, A study of lunar impact crater size-distributions, The Moon 12 (2) (1975) 201–229. doi:10.1007/BF00577878.

[34] E. Shelhamer, J. Long, T. Darrell, Fully convolutional networks for semantic segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (4) (2017) 640–651. doi:10.1109/TPAMI.2016.2572683.

[35] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: N. Navab, J. Hornegger, W. M. Wells, A. F. Frangi (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, Springer International Publishing, Cham, 2015, pp. 234–241.

[36] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A. C. Berg, SSD: Single Shot MultiBox Detector, in: B. Leibe, J. Matas, N. Sebe, M. Welling (Eds.), Computer Vision – ECCV 2016, Springer International Publishing, Cham, 2016, pp. 21–37.

[37] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, MobileNetV2: Inverted residuals and linear bottlenecks, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520. doi:10.1109/CVPR.2018.00474.

[38] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, P. Dollár, Microsoft COCO: Common Objects in Context, arXiv [Preprint] arXiv:1405.0312 (2015).

[39] F. Chollet, Xception: Deep learning with depthwise separable convolutions, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1800–1807. doi:10.1109/CVPR.2017.195.

[40] Lunar reconnaissance orbiter camera (lroc) database, `http://lroc.sese.asu.edu/`, accessed: 19-02-2022.

[41] A. Comellini, D. Casu, E. Zenou, V. Dubanchet, C. Espinosa, Incorporating delayed and multirate measurements in navigation filter for autonomous space rendezvous, Journal of Guidance, Control, and Dynamics 43 (6) (2020) 1164–1172. doi:10.2514/1.G005034.

[42] H. L. Alexander, State estimation for distributed systems with sensing delay, SPIE Data Structures and Target Classification 1470 (1991) 103–111.

[43] T. D. Larsen, N. A. Andersen, O. Ravn, N. K. Poulsen, Incorporation of time delayed measurements in a discrete-time Kalman filter, Proceedings of the IEEE Conference on Decision and Control 4 (December) (1998) 3972–3977. doi:10.1109/cdc.1998.761918.

[44] A. Konopliv, A. Kucinskas, W. Sjogren, J. Williams, A. Binder, L. Hood, The gravity field of the moon from the lunar prospector mission (1998).

[45] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: 3rd International Conference for Learning Representations, San Diego, CA, 2015. arXiv:1412.6980.

[46] S. J. Robbins, A New Global Database of Lunar Impact Craters > 1–2 km: 1. Crater Locations and Sizes, Comparisons With Published Databases, and Global Analysis, Journal of Geophysical Research: Planets 124 (4) (2019) 871–892. doi:10.1029/2018JE005592.

[47] L. M. Downes, T. J. Steiner, J. P. How, Lunar Terrain Relative Navigation Using a Convolutional Neural Network for

Visual Crater Detection (2020) 4448–4453arXiv:2007.07702, doi:10.23919/acc45564.2020.9147595.