POLITECNICO DI MILANO

SCHOOL OF INDUSTRIAL ENGINEERING AND INFORMATION

DEPARTMENT OF AEROSPACE SCIENCE AND TECHNOLOGY

Master of Science in Space Engineering

Vision based hazard maps generation for safe Moon landing: algorithms enhancement, fine tuning and experimental facility setup

Advisor: Prof. Michèle Lavagna

Author: Marco Ciarambino

Academic Year 2014-2015

To my Mother, my Father
and my Sister.

Abstract

The capability to land autonomously and precisely is a key factor for future missions. Scientific interesting regions are often located in hazardous terrains, and it is not always possible to define a landing area in advance, for example when dealing with asteroids or comets, that are too small and too far to be observed with such a great accuracy. Therefore, an autonomous system able to select a safe landing area in a particular interesting region increases enormously the mission flexibility. In this thesis, an hazard detection system based on a single camera working in the visible spectrum is presented. Artificial Neural Networks are exploited to process the camera outputs and landing site is selected with a dedicated algorithm. A routine to validate the landing sites chosen has been developed, in order to assess the hazard detection system level of safety through the comparison with the ground truth solution, developed with NASA LRO mission elevation data. Various design iterations are tested and compared to choose the best definitive hazard detection system. At the end, the design of a facility dedicated to test vision based autonomous navigation and hazard detection systems currently under development at PoliMi is presented.

Key words: Hazard detection, autonomous landing, artificial neural networks, experimental facility

Sommario

La capacità di atterraggio preciso e autonomo è fondamentale per le missioni spaziali future. Regioni scientificamente interessanti si trovano spesso in aree pericolose, così come non è sempre possibile scegliere un luogo di atterraggio in anticipo, ad esempio quando si tratta di corpi celesti lontani e di piccole dimensioni, come comete ed asteroidi, la cui superficie non può essere analizzata con sufficiente precisione. Dunque, un sistema di rilevamento del pericolo capace di scegliere un luogo di atterraggio in autonomia aumenta enormemente la flessibilità della missione. In questa tesi viene presentato un sistema di rilevamento del pericolo (hazard detection system) basato su una monocamera che opera nello spettro del visibile. Vengono sfruttate reti neurali artificiali per processare le immagini provenienti dalla camera e scegliere un luogo di atterraggio sicuro. Questo viene validato da una routine apposita, la quale controlla se il luogo scelto è sicuro anche sulla mappa di pericolosità reale, sviluppata con dati altimetrici della missione LRO della NASA. Varie iterazioni nel progetto del sistema vengono paragonate alla versione finale in termini di prestazioni. Infine, viene esposto il progetto di una facility attualmente in sviluppo destinata al test e alla validazione di sistemi di navigazione autonoma e rilevamento del pericolo basati su sensori ottici.

Parole chiave: Rilevamento pericolo, atterraggio automatico, reti neurali artificiali, laboratorio sperimentale

Acknowledgments

I would like to thank Prof. Michèle Lavagna to have supervised my work with her great experience, giving me the chance to develop such an interesting research. My deepest gratitude to Paolo Lunghi, Ph.D. candidate who patiently advised me in every situation sharing his knowledge and his time, making this work possible.
A special thanks to all my friends at PoliMi, in particular Stefano, Matteo, Riccardo, Giulio, Claudio, Andrea, Vincenzo, Amedeo that lightened the hard journey towards the end. Also, an honorable mention to my lifelong friends, which fortunately are too many to be cited in a single page.
My greatest gratitude to my parents, that not only allowed and supported me in any of my choice, but most of all because the education they gave me, which I have been able to understand only in recent times. The same goes for my sister, who made my life easier paving the way in front me, and to my aunt and Antonio, whose door is always open for me. This thesis is primarily because of them.
Eventually, I want to thank my girlfriend, Benedetta, who has the special power to make my life colorful and to make me feel happy.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

THE most important topics concerning this work are hereby briefly presented. This chapter discusses the concept of autonomous landing capability for spacecraft, thematic which this thesis deals with through the creation of hazard maps to support the landing site selection.
To understand the choices made by the author, it is also necessary to present the state of the art for what concerns navigation and landing related technologies like active and passive sensors. Moreover, an overview of the current european facilities dedicated to the simulation of vision-based landing systems will be introduced.

1.1 Autonomous planetary landing

Safe and precise landing are a mandatory characteristics of space systems in future missions. Nowadays, some of the most advanced space probes include the capability to land semi-autonomously on the surface of a celestial body, that is typically the Moon, Mars, or a Near Earth Objects (NEO). Semi-autonomous landing refers to the ability of a lander to execute in autonomy a series of programmed commands that have as scope a safe landing, integrating on board sensors informations with the predefined mission profile.
Speaking about the **Moon**, for example, it is possible to mention the recent *Chang'e 3* mission. Featuring a lander and a rover, it is the first chinese mission to have performed a lunar soft landing. During the last phases of the powered descent to the surface, the Descent Camera is exploited to avoid obstacles on the terrain and select a low hazard

landing site [1]. Russian led *Luna* missions have a long heritage in lunar landers and rovers, starting with the first human object hitting the Moon with Luna-2 in 1959. In particular for the planned missions Luna-25 and Luna-27, an hazard detection and an autonomous navigation systems could be provided by ESA, such as part of the scientific payload [2]. India has its *Chandrayaan* program, in which a first orbiter (Chandrayaan-1) is already been successful. Chandrayaan-2, a lander capable of soft landing, is planned to be launched in 2017. It will exploit hazard avoidance and autonomous navigation through cameras [3].

Mars has been object of numerous mission and studies in the past, but many more are still to come. Probably the most famous mission is the *Mars Science Laboratory (MSL)* by US space agency NASA. Active guidance systems have been adopted to increase the landing precision to 20 km [4]. It can be considered a first iteration towards the autonomous landing of future spacecraft: no hazard detection systems were available on board, and the rover Curiosity landing site has been *a priori* determined, without the possibility of a real-time retargeting by the spacecraft. Quite interesting is the first use of the *skycrane*, that is a structure devoted to soft-land the rover to the Mars surface through ropes, hovering distant from the terrain. In any case, the automation of this landing system is limited to the execution of predetermined phases. ESA's *ExoMars* program intends to deliver an orbiter, a lander and a rover to Mars, with two separate missions: ExoMars 2016 (orbiter and lander), Exomars 2018 (rover). Also in this case, there is not a fully autonomous landing system capable of real-time terrain analysis and retargeting, but the landing site is predetermined with a large error ellipse: 104 km $\times$ 19 km for ExoMars 2018 [5]. Another NASA mission, planned for launch in 2020 is *Mars 2020*. It is a rover based on Curiosity, but with different scientific instruments on board. The atmospheric entry and landing systems are the same as the one exploited for MSL, that are possible to be considered as semi-automatic. An interesting project is ESA's *Phootprint*, a Phobos sample return mission study to be launched in 2022 devoted to investigate which process is most likely the one that formed the martian moons. Four segments constitute the mission: Propulsion Module, Lander/Orbiter, Earth Return Vehicle, Re-entry Capsule [6]. The lander itself is responsible to the selection of the sample site

when inserted in orbit around Phobos.
A great scientific interest comes also from **asteroids** and **Near Earth Objects (NEO)** in general. The *Asteroid Impact Mission*, for example, is an ESA project devoted to the exploration of the binary asteroid *Didymos* and to assess planetary defense capabilities. Together with the *Double Asteroid Redirection Test (DART)*, it composes the *Asteroid Impact and Deflection Assessment* (AIDA). DART is a single impactor spacecraft that hits the smaller member of the binary asteroid to change its orbital period. AIM probe determines the momentum and other physical characteristic transfers due to DART impact. Another important project is *OSIRIS-REx* mission, designed to return samples from the carbon-rich Bennu asteroid to study the origin of carbon-based life. To pick a sample from the surface, the spacecraft features Touch-And-Go Sample Acquisition Mechanism (TAGSAM) sample collector: a burst of nitrogen gas pushes the asteroid surface regolith inside the sampler's chamber. Touch and go was chosen to eliminate the risks deriving from a landing onto an asteroid. The selection of the candidate sample site is performed through topographical mapping with OSIRIS-REx LASER Altimeter (OLA), a scanning LIDAR [7].
When talking about landings on objects in space, ESA's Rosetta mission must be taken in consideration. Along with its lander Philae, it is investigating the comet 67P/Churyumov–Gerasimenko. It is the first mission in human history to rendezvous with a comet, escort it as it orbits the Sun, and deploy a lander to its surface [8]. Philae landing site have been chosen by the Landing Site Selection Group (LSSG) after receiving images from Rosetta onboard OSIRIS and NAVCAM cameras. Landing sequence has been loaded to the Philae lander after it has been computed on Earth.
Due to the telecommunication delays between the spacecraft and the ground segment, a modern landing system must feature a high level of autonomy too, being able to adapt and change landing site during the landing phase, almost to the touchdown. Various reasons could drive a landing site to be changed during the descent: a failure in an on-board system, dictating an immediate landing in the nearest and safest possible site, a scientific interesting spot detected real-time, the need to avoid one or more hazardous areas in the region designated to land on. In any case, the on board systems must have the authority and the ability to detect unpredicted events and to compute an

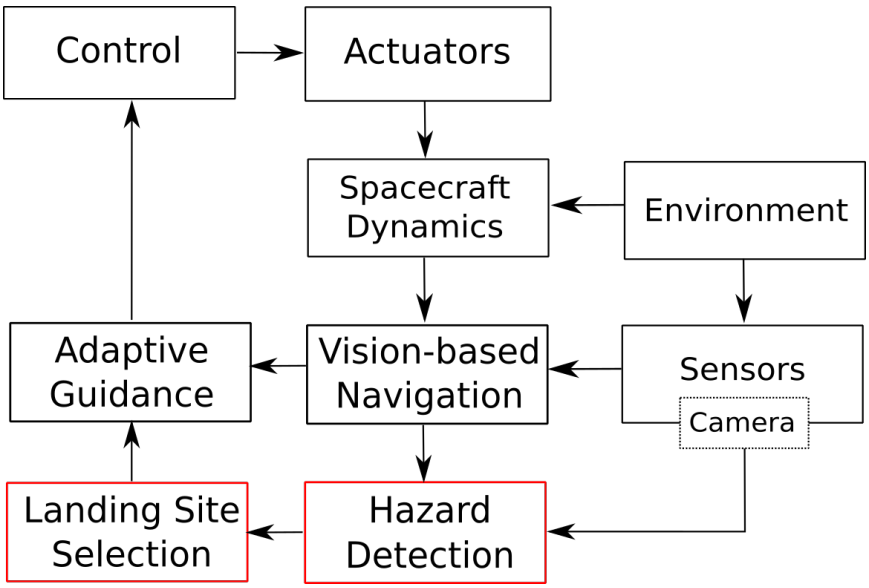


Figure 1.1: Functional architecture for an autonomous landing system. Red boxed subsystems are studied in this thesis.

adequate solution autonomously. In literature, it is common to refer to *Autonomous Guidance, Navigation and Control (AGNC)* when alluding to an on board system devoted to safely maneuver the spacecraft with a certain logic, especially towards a safe touchdown with the planetary surface. It is a closed-loop set of subsystems (Fig. 1.1) designed to be able to scan the area around the landing site, to check if it is possible to reach for the nominal target site with the required level of safety and, if not, to search for an alternative spot that meets the landing requirements. In such an occurrence, a new landing trajectory towards the updated target landing site should be calculated, followed by the execution of a divert maneuver. The capability to select another landing site whenever it is requested is provided by a *vision-based Guidance, Navigation and Control (GNC)* chain, composed by three main components:

- Adaptive Guidance (AG) system;
- Hazard Detection (HD) system;
- Vision based Navigation (VN) system.

In addiction, a *landing site selection algorithm* selects the most suitable landing spot on the surface.
A quick review of the subsystems is given:

Sensors. They supply the input to the whole system. Usually, inertial systems (IMU), passive optical devices (star trackers, cameras), active optical tools (LIDAR), active radio-waves equipments (RADAR) are exploited. Typically, raw input is processed to extract, to enhance, to transform data useful to the next blocks, i.e. **vision based navigation** and **hazard detection**.

Visual based Navigation (VN). This subsystem reconstructs the dynamic state of the spacecraft through the informations coming from optical sensors, that is merged with outputs from traditional sensors, like Inertial Measurement Units (IMU) and altimeters, to provide relative or absolute navigation.

Hazard Detection system (HD). This subsystem transforms the input image from the camera into a *hazard map*, in which each pixel represents the hazardousness of the corresponding input subspace, taking into account slopes, shadows, terrain roughness. Through such map, the landing site selection algorithm computes the most suitable spot on the surface for the spacecraft to land on.

Adaptive Guidance (AG). Once the target landing site is known, AG subsystem computes the trajectory to reach for the target, taking into account current lander dynamics and constraints, such as fuel consumption, target visibility and lander control authority.

Control system and actuators. The control system determines the magnitude of the command to the actuators comparing the current kinematic and dynamic parameters with the ones computed by the AG to reach for the target landing site.

This thesis, developed at Department for Aerospace Science and Technology of Politecnico di Milano, focuses on the detection and

the avoidance of hazardous areas on the surface of celestial bodies, with particular attention to our natural satellite the Moon, making possible for a guidance and navigation control system to drive the spacecraft to a safe landing site. Specifically, a single camera coupled with an artificial neural network is exploited to provide not only a robust and reliable, but also a cost effective and computational light hazard detection system.

1.2 Hazard Detection and Avoidance

1.2.1 Definition of hazard

Talking about rocky celestial bodies, danger is represented by morphological features that make difficult or impossible for the spacecraft to land and to operate safely and compliant with the mission requirements. Such hazardous characteristics are *craters, rocks and slopes*. Obviously, danger depends not only on the particular terrain feature

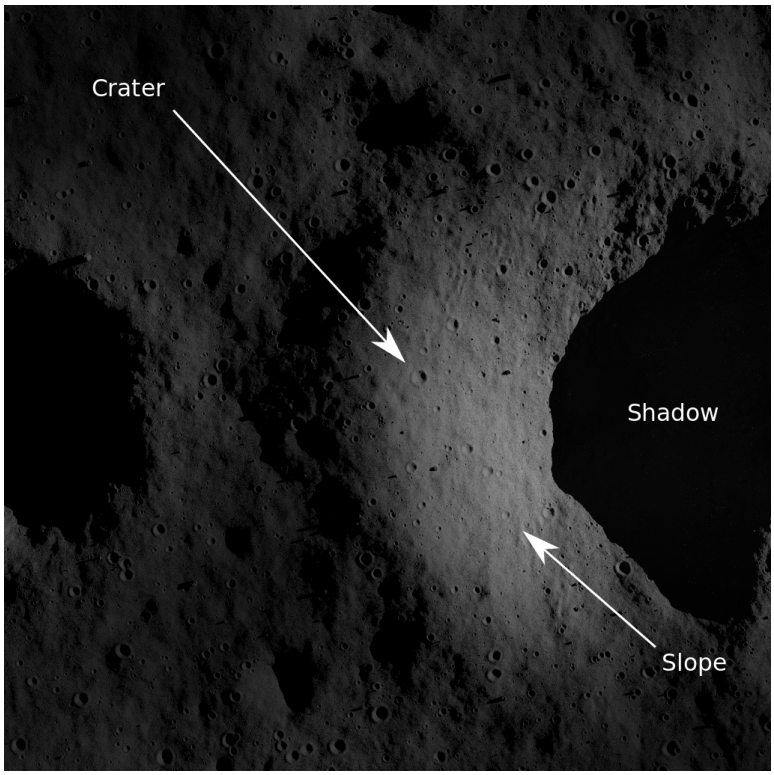


Figure 1.2: Surface features

itself, but also on the size and the characteristics of the lander: as a huge spacecraft would see a certain rock like a little stone, another one could not be able to operate or even worse be damaged with the same conditions. Or a particular slope might be well handled or not depending on legs of the lander. Thus, it is necessary to assess the *hazard index* taking into account both the planet surface and the spacecraft. Moreover, dealing the present work with hazard detection through a camera operating in the visible spectrum, shadows must be considered hazardous, being *de facto* areas outside the sensor field of view.

1.2.2 Landing phases and Hazard Detection and Avoidance

Generally speaking, a landing maneuver for planets with no atmosphere like the Moon, can be divided into four sub-phases:

- **Deorbit burn and coasting phase**: the spacecraft is usually considered to begin the landing from a low altitude parking orbit or from an hyperbolic trajectory in case of direct landing from a transfer orbit. Performing a first engine burn, the spacecraft is injected in an elliptical transfer orbit with periselene located at about 10 - 20 km height.

- **Main brake**: performed at the periselene of the transfer orbit, main brake phase occurs when the spacecraft main propulsion system is activated at max thrust to decrease as fast as possible the most of the orbital velocity. At the end of the maneuver the spacecraft performs a pitch-up maneuver and the hazard detection system starts to operate.

- **Approach**: between 2500 and 1500 m in altitude main engine thrust is reduced to be able to maneuver. Hazard Detection and Avoidance tasks take place in this phase: the target landing site is finalized, and one or more divert maneuvers can be performed.

- **Terminal descent**: when it is reached the vertical of the designated landing site, at about 50-20 m height, a constant speed descent at about 3 m/s is maintained until the touchdown.

As stated, Hazard Detection and Avoidance deals with the approach phase. If necessary during the descent, such maneuver would follow a process similar to the following:

1. Hazard Detection and Avoidance (HDA) maneuver begins between 2500 and 1500 m in altitude at the so-called **HDA High Gate**, after performing pitch-up maneuver, the lander is supposed to have an almost vertical attitude with a vertical velocity and an horizontal velocity respectively in the order of 15-30 m/s and 10-20 m/s.

2. **Large Scale Hazard Avoidance Maneuver** is accomplished:

   (a) HDA system scans the current landing area in input from the camera and build the relative *hazard map*. An algorithm selects the most suitable landing site, that is not depending only on the pure hazardousness threshold, but it must take into account many other parameters, such as fuel consumption needed to reach for it and distance from an eventual mission nominal landing site.

   (b) Landing trajectory is computed taking into account vehicle dynamic characteristic, attitude, current trajectory, fuel on board. Thus, the consequent diversion maneuver is commanded. The target point for the trajectory is defined **HDA Low Gate** and it is located between 600 and 350 m on the vertical with respect to the landing site. Velocities drop to 5-10 m/s (vertical) and below 5 m/s (horizontal). Attitude is vertical with engine nozzles pointing perpendicular to the terrain.

3. The **Small Scale Hazard Avoidance Maneuver** is performed:

   (a) the HDA system scans again the landing area, computing the relative *hazard map*. If it is necessary, algorithms update the landing site with the most fitting one.

   (b) Up to date trajectory is computed and the new diversion maneuver is commanded. Target point, called **Terminal Gate**, is now located just few meters above the selected landing site.

4. **Terminal Descent**: With a vertical attitude and a null horizontal velocity, the last phase consists in a constant vertical velocity descent until the touch down.

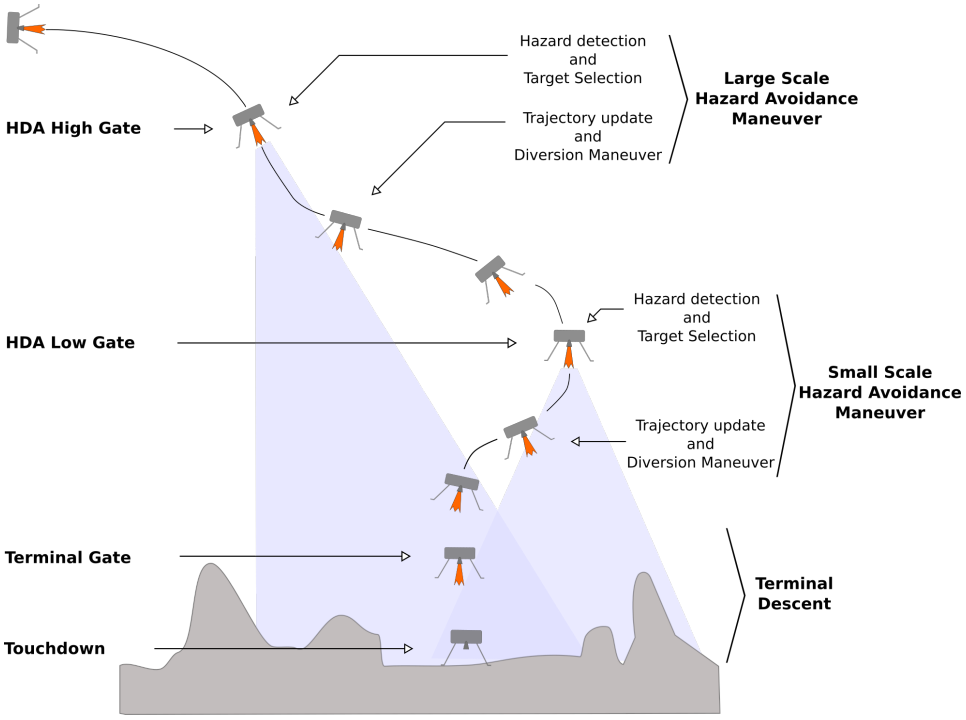


Figure 1.3: Hazard Detection and Avoidance during the Landing Phase in celestial bodies without atmosphere.

1.3 State of the art

1.3.1 Sensors for autonomous landing systems

RADAR altimeter The RAdio Detection And Ranging (RADAR) altimeter exploits reflection of electromagnetic waves onto a surface to compute the relative distance. In particular, timing the interval between sending of the signal and receiving the echo, it is possible to understand the distance between the transmitter and the ground onto which the beam has been reflected. They are mounted on landers in order to have a precise altitude measurement, a fundamental parameter to know when it is necessary to turn on the propulsion system to

perform a correct and precise landing maneuver. Typically, their resolution allows them to compute a reliable altitude value below 2000 m [9]. Because of the low spatial resolution of these devices, they cannot be used to detect morphological features of the terrain.
The same approach of the RADAR altimeter can be exploited to identify obstacles on the planetary surface: reducing the waves beam emitted, it is possible to increase the spatial resolution, but of course a lower area is scanned. Thus, the Millimeter Wave RADAR [9] (MMW), must embed a device able to change the beam direction, in order to scan a sufficient terrain area. In space applications, due to tribological and maintenance issues, mechanisms are preferably avoided, hence it is necessary to change the direction of the waves beam without mechanically rotating the antenna. An example of such an approach is represented by *reconfigurable-plasma refraction antenna* [9].

LIDAR Following the same physical principle of the RADAR, the LIght Detection And Ranging system, utilizes echoes of a light source onto a surface instead of radio waves. Being light's wavelength much smaller than radio waves', it is possible to detect much smaller obstacles or discontinuities of the terrain profile. On the other hand, such a high frequency of the wave can yield to false values when encountering suspended particles in atmosphere with dimensions of the order of the wavelength of the beam emitted.
Through a such concentrated beam, a LIDAR can be exploited as an altimeter from much higher heights with respect to a radio waves based counterpart [10, 11]. With respect to a camera operating in the visible spectrum, the LIDAR has less limitations due to its intrinsic active nature. Even though it is a very promising technology as navigation equipment, technical complexity, high price and electrical power needs [12] are still to be overcome for a common implementation on space missions. Nevertheless, through technological and economical improvements, future mission are expected to rely heavily on such instrumentation.

Camera Cameras are passive optical devices, usually working in visible spectrum. The core component is the Charge-Coupled Device (CCD): a small doped silicon board highly photosensitive divided into

pixels. As photons hit a pixel, it converts the intensity of the incoming light beam to electrons, depending on the amount of photons received. It is thus possible to reconstruct the image computing the number of electrons per pixel. Cameras can present different Field of Views (FoV) and in general horizontal FoV is different from vertical FoV to have a privileged direction. It is a mature technology in both terrestrial and space applications with respect to LIDAR, while it is cheaper and less power consuming than RADAR. It is also widely used because of the low power consumption and the ease of use, due to the fact that the output of a camera is already the image itself. More, the *image depth* is an important parameter that represents the intensity resolution that is possible to reach. Number of pixels present on the CCD yields the image resolution of the output picture. The higher the resolution, the more detailed picture is possible to obtain. On the other hand, more memory and computational power is needed to process and storage images. The absence of any moving part makes the camera a very reliable sensor.
Anyway, due to its passive intrinsic nature, camera performances are affected by the different light conditions in the various operating situations. Nevertheless with a pure single camera configuration, being the output image bi-dimensional, it is not possible to directly obtain an estimation of the depth information, with consequent problems in computing steepness and differences in height of the planetary surface.

1.3.2 Automatic Guidance and Navigation Control Systems

Many architectures to provide a AGNC have been developed and are currently in progress by both space agencies and private companies. Generally speaking, they integrate measurements from multiple sensors, exploiting inertial and relative navigation techniques. The reader should note that the following systems –with the only exception of ALHAT– are mainly *navigation* systems, with very low capabilities in terms of hazard detection. Indeed, HD is probably the less investigated area of the whole AGNC chain. An overview of the most notable architectures devoted to planetary navigation is presented hereafter.

SINPLEX The Small Integrated Navigator for Planetary EXploration (SINPLEX) [13] is an autonomous navigation architecture in design and test phase. Developed by a consortium headed by the DLR institute, it is designed to be lighter than current counterparts and to yield high precision landing phase on planets and small celestial bodies and/or rendezvous with other spacecrafts [14]. Low mass is obtained through a miniaturization and integration of the various components. SINPLEX can count on various sensors:

- LASER altimeter
- IMU
- Star tracker
- Navigation Camera (NavCam) with dedicated image processing unit

Both Crater Navigation and Feature Tracking are available for known surface bodies. On lunar landings, it is designed to yield a landing precision of 100 m at 1σ, while 1 m on asteroid landings [15].

ALHAT The Autonomous Landing and Hazard Avoidance Technology is in active testing phase at NASA [16]. Its main focus is to provide a AGNC system that needs almost null *a priori* knowledge on the terrain which the spacecraft has to land on [17]. It packs an hazard detection and avoidance system, navigation and guidance support. Main component is the *Flash LIDAR*, that allows ALHAT to be able to guide the spacecraft in any light condition through Terrain Relative Navigation (TRN), HDA, and Hazard Relative Navigation (HRN), coupled with IMU, star tracker, altimeter, doppler velocimeter. It aims a landing precision within tens of meters if lunar navigation assets are present and/or lunar map of the area are known by the system. Otherwise, landing precision falls down to less than 1 km as magnitude [18].

Lion Co-funded by ESA, ASTRIUM and ONERA, Lion is an Absolute Visual Navigation system based on surface features matching [19]. Due to this, it exploits previous mission data through an *offline process* [20], in which the system prepares the various landmarks to be used

during the autonomous navigation process through an Harris Laplace feature extractor (see [21] for a brief description of image features extractors). Then, the *online process* [20] is activated to compute the estimation of the spacecraft state. During features matching, among the various matches, three sets of recognized features are used for position, velocity, attitude computation, discarding the others through a RANSAC algorithm [19].

1.3.3 Hazard Detection techniques

The vast majority of HD system exploits the creation of hazard maps to provide the most suitable target landing site. Various parameters, such as maximum terrain slopes and rocks height, together with the spacecraft characteristics, contribute to the creation of such hazard maps. The notable difference among HD system is about *how* they are able to create such maps. The most relevant techniques are presented.

Mono-camera Mono-camera systems are cheap and reliable, and their output is already in form of image, thus it is easily processable. On the other hand, they are not able to directly yield depth information and HD systems performances based on cameras are strongly influenced by light conditions during operations. To generate the hazard map, HD systems with a single camera relies typically on *shadow-based* vision techniques [22, 23] or on *shape from shading* algorithms, that is able to reconstruct depth through intensity of the reflected incoming light beams [24, 25]. An example of single camera HD system proposed by ESA exploits shape from shading techniques to compute slopes, histogram thresholding to isolate shadows, standard deviation of the pixels intensities to measure surface roughness [26]. Such three maps are then integrated in a total hazard map on which landing candidates that respect lander and safety requirements are computed. Another approach with single camera is proposed in [27]. Here, an image segmentation algorithm through local intensity clustering identifies regions as safe or unsafe. It can be considered an extension of more conventional shadow thresholding based image processing techniques. A completely different image processing technique that exploits a single camera is the so called *structure from motion* [28]. Used often in terrestrial vehicular applications [29] for ac-

tive safety systems, it is a set of techniques that allows to reconstruct the shape of the object by successive frames, knowing the kinematics of the camera during the image frame acquisitions. It constitutes the hazard detection system's core in [30], where an autonomous helicopter with a camera on board is able to compute slopes and surface roughness: through the boolean version of such two maps and taking into account the helicopter footprint, it is possible to compute the suitable landing sites.

Stereo camera Stereo camera techniques exploits the difference in perspective of the acquired images to compute depth. The larger the baseline, the more precise third dimensional information is calculated. An algorithm for hazard detection is implemented in [31]: an algorithm dedicated to the slope estimation produces an *elevation map* with the two cameras and fits the points of such map to compute slopes. A second algorithm, always based on stereo vision, detect rocks: surface plane is thresholded to discard noise and negligible features. Deviation of the remaining features is computed, and averaging the highest points in the regions, the height of the rock is computed. Also a shadow based rock detection algorithm is present, to be used especially at higher altitudes.

LIDAR The most relevant hazard detection performed through LIDAR is ALHAT system. Clearly, due to its active nature, it does not relies on external light sources, therefore the hazard map is based on surface roughness, slopes, distance from the nearest hazardous feature. Also in this case, the three maps are generated one by one and then integrated in the final hazard map. Specifically they are generated through a *flash LIDAR*, that creates a 128×128 px image with each pixel corresponding to intensities of the return laser pulse. The maximum of such intensities is registered and its correspondent time of flight is multiplied by speed of light over two in order to obtain the range for that pixel. Using a perfect camera projection for the computed laser rays, a 3D cloud of points is obtained in LIDAR's reference frame. Transforming data into Universal Transverse Mercator frame, it is possible to obtain a computed Digital Elevation Model (DEM) of the surface. Such calculated maps are then correlated with the onboard DEMs, computed through previous missions, and a Terrain Rel-

ative Navigation (TRN) algorithm estimates spacecraft position [32]. Another example is represented by [33], in which hazard maps are exploited to compute landing sites onto Mars surface. In such work, not only the HD system, but also a terrain simulator has been developed to be able to tailor every system before laboratory or real world tests.

1.3.4 Facilities

An experimental facility represents a mandatory step to the integration of a system in actual aerospace operations. The laboratory environment allows the development and the validation of breadboard, engineering, protoflight and flight models increasing the Technology Readiness Level (TRL) of the system under development. Moreover, using actual space-hardware, it is possible to assess the real-time performances and to understand the various coupling effect between the constituents of the system.

TRON The Testbed for Robotic Orbital Navigation is a facility located at DLR in Bremen dedicated to the simulation of lunar landing phase, in particular to provide the environment to qualify optical navigation sensor breadboards to TRL 6 [34]. It consists in a dark room[1], with two walls devoted to host the lunar dioramas. A KUKA 7 DoF robotic arm is mounted onto a rail guide to be able to simulate navigation of the spacecraft, whose sensors are mounted onto the end effector of the arm. Another rail guide drives the 4 DoF illumination system that simulates sunlight color temperature onto lunar surface. Note that TRON is designed with in mind the possibility to serve as testbed not only for cameras, but also LIDAR and active devices in general. That is why it has been opted for a 3D diorama [34]. The manufacturing process of the diorama itself is made in house by DLR. Milling process and polyurethane foam have been selected to provide the correct accuracy needed to have a reliable navigation simulation environment. Matte powder based coating for the diorama assures optical requirements of the model surface to be coherent with lunar soil ones. Scale factors to represent space environment varies from

[1]Necessarily covered with black matte fabric to not allow any unwanted reflection to undermine the simulation fidelity.

real scale 1:1 to 1:50000, limited by manufacturing precision and availability of high resolution DEMs [35, 34].

VisiLab VisiLab is a test bench set up to validate vision-based navigation systems. It is located in Noordwijk, The Netherlands at ESA - ESTEC. It is composed by a lunar diorama, a support for the camera to simulate spacecraft trajectory, an illumination system, a calibration framework [19, 20]. It is devoted to the validation of the Lion AGNC system (Sec. 1.3.2). Mock-up specifications for the lunar surface have been taken from the NASA LRO mission data. In particular, Moon South Pole has been selected for the diorama because of its scientific interest [19]. Manufacturing process has been performed at DLR Bremen, the same as the one for the TRON facility. As camera support it has been used a 4 DoF[2] structure, instead of a typical robotic arm [20].

VBNF The Vision-Based Navigation Facility (VBNF) is a test bed and proof of concept for vision-based navigation systems, to increase their on-board autonomy [36]. Located at Thales Alenia Space premises in Italy, it simulates the Entry, Descent and Landing (EDL) phase of a lander. It is composed by:

- a quadricopter drone: endurance of 15 minutes, it packs as payload the camera, the trajectory control board the serial radio for a total flight mass of 2.7 Kg;
- the Tracking system, used to track position and attitude of the drone to the trajectory control workstation;
- the diorama, a huge 8×8 m in scale 1:300 representing some important features of Mars surface;
- the simulator/image processing workstation, used to validate the mock-up features.

An important remark: the VBNF does not simulate the dynamics of the spacecraft, it is a facility designed to validate *vision based* modules, components in charge of acquiring and processing images during the Entry Descent and Landing (EDL) phases.

[2] three translational direction, rotation about pitch axis

PLGTF The Precision Landing GNC Test Facility (PLGTF) is an European Space Agency (ESA) outdoor facility developed with the purpose to validate GNC techniques to be used on future Moon and Mars missions. Its core component is a drone, whose flexibility allows to mount camera and LIDAR based GNC systems. PLGTF is capable to emulate a lander dynamics during the powered descent phase in atmospheric and non-atmospheric planetary environments [37]. The facility is composed by:

- *Flight segment*. An Unmanned Aerial Vehicle (UAV) helicopter based on the Schiebel CAMCOPTER S-100, adequately modified to suit the PLGTF needs of payload (camera and LIDAR);
- *Ground segment*. Ground stations, test site, and logistic infrastructures compose the ground segment. Ground stations are devoted to control the module necessary to validate the experiment under test (M3 Service Module), measuring and estimating the dynamic behavior of the drone and to control the drone itself simulating lander descent. Test site has been selected in the Morocco desert, near Zagora city, because of the terrain resemblance with Mars' and Moon's. Both lunar and martian descent trajectories have been simulated at the drone manufacturer premises, showing the feasibility and the validation capabilities of the facility. For details, see [37].

HOMER Airbus Space and Defense HOver ManoEuvRe (HOMER) is a demonstrator capable to hover and to perform soft landings, designed to be a landing platform for planetary exploration and an agile platform for space mobility [38]. It is currently in advanced test phase, with first hover ground test passed. It is not designed exclusively as test-bed for hazard detection or navigation, but its flexibility guarantees the operations of HOMER from space debris mitigation to autonomous planetary landing, tailoring the payload for every different mission requirement.

Morpheus NASA's Morpheus is a testbed for validate and test planetary landing systems. Its primary payload is ALHAT, the guidance system described in Section 1.3.2. It is composed by a liquid methane/liquid oxygen propelled rocket and it is capable to land vertically and

autonomously. Among the various test, on the 29th of May, 2014, Morpheus, driven by ALHAT, accomplished its first safe landing in dark [39]. Before Morpheus, a UH-1H 'Huey' helicopter has served as test-bed for 12 tests of the ALHAT system.

1.4 Development tools used

The first hazard detection system design iteration, called ORACLE+, was developed totally in Matlab® environment, with the hardware-in-the-loop dedicated program written in C++. Neural networks have been trained through the Neural Networks Toolbox in Matlab® and then exported to C++.
Nevertheless, it has been decided to start a migration towards Free and Open Source Software (FOSS) tools also for development, in order to be independent from proprietary licenses and to be as cost effective as possible. Such a shift has began during the development of the ultimate version of the hazard detection system, and currently, only the creation of the images and corresponding ground truth hazard maps (Chapter 4) is still based on Matlab®.
Many possibilities were open to choose the programming language, the various libraries to train the networks and so on. It has been decided to trade-off various features that languages have to offer. In particular, Python and C++ have been selected as programming languages. The first is easy to use and has a huge community driving excellent libraries development for scientific and engineering applications, without mentioning the great compatibility that SciPy libraries have with the Matlab® file format. It has been chosen as a development language to perform all the various accessory tasks that are not part of the hazard detection program itself that goes in the hardware, such the validation of the results, performances computation, creation of the datasets and so on. Instead, whenever high performances were requested, C++ has been preferred, thanks to its tremendous computational speed. Hence, training of the network and the actual hazard detection program are written in C++.
For image processing, OpenCV 2.4.10 [40] libraries have been used in both C++ and Python thanks to the already present bindings. Neural network training is performed exploiting the Fast Artificial Neural Networks (FANN) libraries [41]. Many others machine learn-

ing libraries have been tried, also in Python environment, but none of them was direct to use, flexible and computationally efficient as FANN. C++ compiler used is GNU's g++ 4.9.2.

1.5 Thesis objectives and structure

This thesis has two main objectives:

- to design an hazard detection system able to select a safe landing site on the surface of the Moon using a single camera. It must possess the following requirements:
  1. At least two safe landing sites shall be identified in case a re-targeting is requested;
  2. At least 90% of the landing sites found by the hazard detection system shall be true positives, in particular the target landing site and the backup site shall be true positives;
  3. The hazard detection system shall be computationally light enough to operate real time on space qualified hardware.
- to design a facility to validate the hazard detection system proposed.

The work proposed has the following structure: in Chapter 2 a presentation of the main tool exploited to create the hazard detection system, the Artificial Neural Network, is given. A short section dealing the ORACLE+ hazard detection system, by which this thesis is inspired, is proposed in Chapter 3. Chapter 4 explains how ground truth images and hazard maps are artificially crafted to form a complete dataset to train and test the hazard detection system. Chapter 5 analyzes the definitive architecture and the most significant design iterations of the hazard detection system, to compare their performances in Chapter 6. Then, Chapter 7 presents the facility design, to conclude the thesis in Chapter 8, where conclusions and future work suggestions are given.

Chapter 2

Artificial Neural Networks

THIS chapter introduces the reader to a branch of the vast world of machine learning: the Artificial Neural Networks (ANN). In particular, Feedforward ANN are the principal tool utilized to develop the research proposed. Moreover, a very concise introduction about another kind of neural networks, the Self-Organizing Maps (SOM) will be presented, due to their presence in the early development stage of the work subject of this thesis.

2.1 Feed-Forward Artificial Neural Networks

2.1.1 From the Biological Neuron to the Multilayer Neural Network

The human brain is characterized by an enormous amount of neurons: it is estimated a number of 20 billion cerebral cells composes the brain with 60 trillion connecting synapses [42]. Therefore it is a massively parallel system. Actually neuronal events[1] are several order of magnitude slower with respect to silicon-based processes: millisecond order of magnitude against nanoseconds [43]. This is mainly due to the conversion between chemical electrical and again chemical signal occurring at the synapsis. Nevertheless, the human (or mammals in general) cortex is definitely faster in all that processes involving

[1] In neurobiological science, with neuron event it is intended an information processing between a neuron and its neighborhood. Being the brain characterized by a huge level of parallelization, the amount of time spent by an impulse to be transmitted from one neuron to the ones connected is almost the same as the elapsed time between two single cells.

recognition, robust motion control and perception[2]. Again, this is due to its incredible number of synapses between the neurons, that allows such a great computational power. The structure of a biological neu-

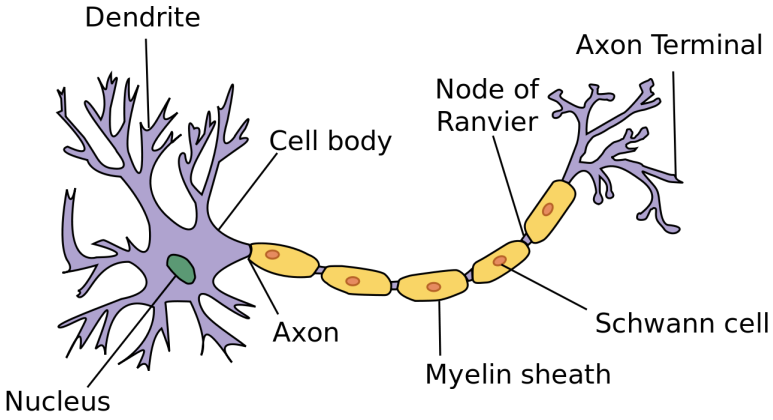


Figure 2.1: The biological neuron.

ron is shown in Fig. (2.1): from the cell body (also called *soma*), *axon* and *dendrites* develops. They are respectively transmission lines and receptive zones of the neuron itself [44]. An axon is much longer and has fewer branches, whereas the dendrites features a great number of short ramifications [45]. Morphologically the first has a smooth surface, while the second is much more irregular. This difference is imputable to the fact that the dendrite is responsible for the majority of the inputs through those dendritic irregularities, called *spines*.
Such structure, pioneered by the neuroscientist Ramón y Cayál [46], inspired the model of the first artificial neural unit: the *Perceptron*, created in 1957 by Frank Rosenblatt at the Cornell Aeronautical Laboratory [47]. Originally, the perceptron was a binary classifier: a map from a vector of real components $\mathbf{x}$ to a binary value y:

$$y(x) = \begin{cases} 1 \text{ if } \sum_i w_i x_i + b > 0 \\ 0 \text{ otherwise} \end{cases}$$

where w_i represents the *weights* of the synapses, b the *bias*, a term not depending on the value of the input and used to shift the threshold of the output. The learning process for a perceptron corresponds to the correct selection of the connection weights and the bias through a

[2]All processes related to the natural and/or artificial intelligence.

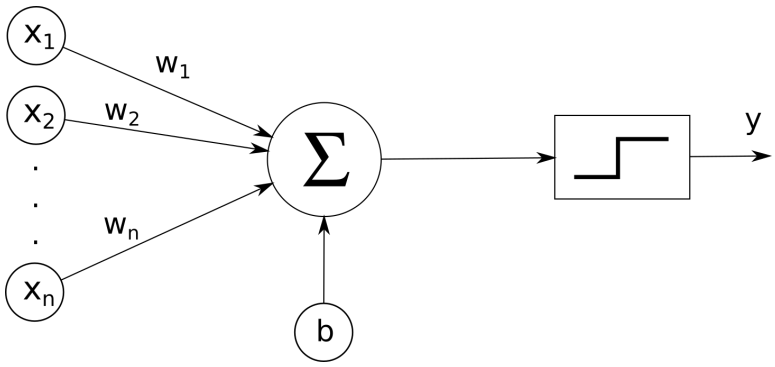


Figure 2.2: Rosenblatt's perceptron.

"training" process, in order to be able to assign coherently the binary output to an input, that in general is not included in the training set. Soon, it became clear that the perceptron, and even a single layer of more than one perceptrons, cannot be trained to classify patterns that are not linearly separable [48], like the XOR problem. Due to this fact, research on artificial neural networks stagnated, up to when it has been recognized that a multilayer neural network with two or more layers can solve also non linear patterns [49].
An artificial neuron represents an evolution of the perceptron model: the general architecture remains quite similar, apart from the presence of a transfer function - the so-called *activation function* - that processes the input signal in any desired fashion, whereas the perceptron outputs binary values depending on the threshold imposed[3]. Thus, it is possible to think of the perceptron as an artificial neuron with a Heaviside transfer function. A single artificial neuron, the core constituent of a neural network, is depicted in Fig. (2.3). It is possible to distinguish, from left to right [44]:

Synapses. They represents the connections with the input of the neuron. Each synapse is characterized by a *weight* that multiply the respective input signal. For a matter of notation, an input signal x_i connected to the i-th synapse attached to the neuron j is multiplied by a weight w_{ji}. Also a bias value b_j is used in order

[3]In general, one could use any kind of activation function, therefore binary or other functions yielding discrete outputs. Typically, continuous function bounded to [-1,1] or to [0,1] output are used because of the necessity to have a differentiable function in the back-propagation training algorithm.

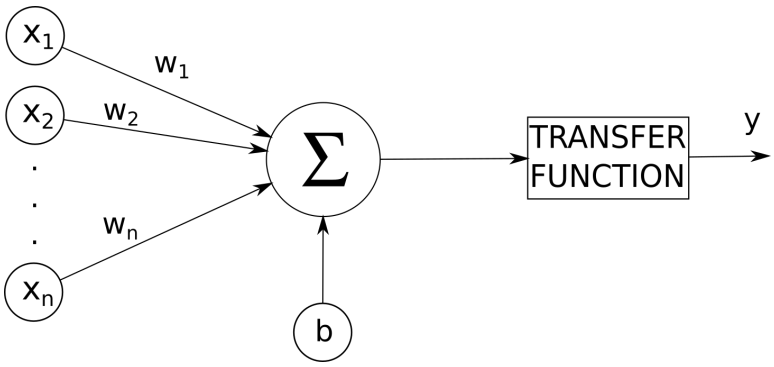


Figure 2.3: Artificial neuron scheme.

to increase or decrease the net input of the activation function.

Summation node. All the inputs converge to an *adder* and are linearly combined.

Activation Function. Used to map the signal after the summation node usually in the range [0,1] or [-1,1]. Most used functions are hyperbolic tangent, linear function, logistic sigmoid (see Appendix A).

Mathematically speaking, a neuron process an information this fashion:

$$y_j = \phi(v_j) \tag{2.1}$$

in which it has been defined the *net output*:

$$v_j = \sum_{i=1}^{m} w_{ji} x_i + b_j \tag{2.2}$$

where m is the number of inputs x_i to the neuron j with output y_j. Each input has a corresponding weight w_{ji}.
It is also very frequent to include the bias into the same expression as the inputs, adding a synaptic weights equal the bias with input signal 1 for $i = 0$:

$$y_j = \phi\left(\sum_{i=0}^{m} w_{ji} x_i\right) \tag{2.3}$$

Hence, it is possible to notice the elementary operations performed by the artificial neural cell.

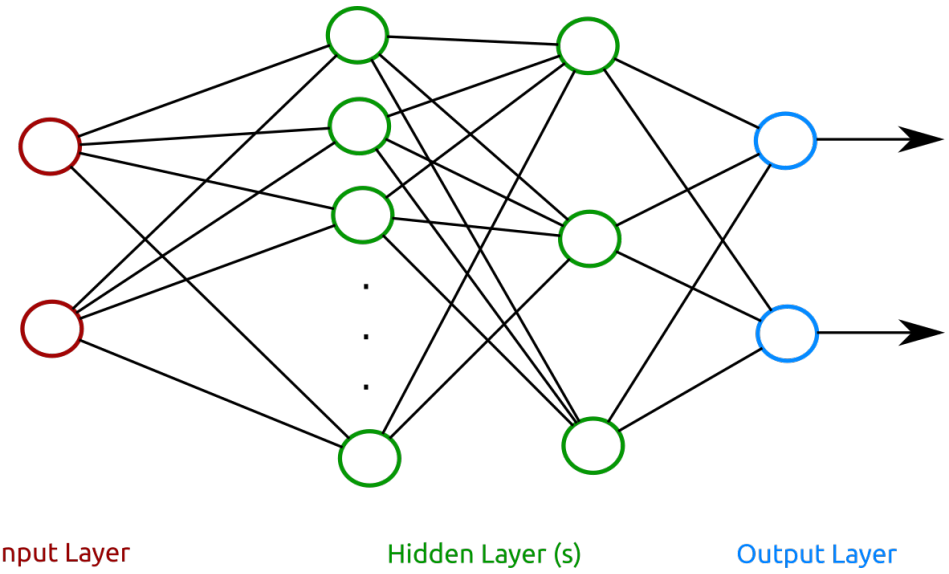


Figure 2.4: Scheme of a Feedforward Multilayer Artificial Neural Network

To be able to extract higher order statistics and recognize more difficult pattern, a *multilayer* artificial neural network is necessary [50, 44]. In particular, a Feedforward artificial neural network features one or more *hidden layers* (Figure 2.4) connected to the input and the output layers. It is common to refer to a *fully connected* network if all the neurons in each layer are synaptically linked to every neuron of the adjacent following layer, otherwise the network is *partially connected*. Structure of a multilayer feedforward network features [44]:

- Neurons' activation functions are *differentiable*. Typically, functions are also nonlinear and bounded to unit value for performances in both training and output quality. Appendix A proposes a list of the most used transfer functions in neural networks.
- The network has one or more hidden layers.
- The network features a high degree of inter-neural connectivity through synaptic weights.

The simplicity of a single neuron hides the complexity of a network: power and strength of ANNs come from the parallel distribution of the architecture and the ability to learn. Learning is intended as the

ability to *generalize*: after a training process, the network, if well trained and well structured, is able to produce a coherent output, given a new input that has been not provided to the net during training. Especially for nonlinear functions, hidden layers neurons act as feature detectors. Their ability is to nonlinearly map the input space in a more profitable "feature space", in which input's informations are more easily distinguishable [44]. Capabilities of a neural network include [44, 51]:

Nonlinearity. Multilayer ANN are intrinsically nonlinear, although single neurons can be both linear and nonlinear due to their activation function. Being the great majority of problems nonlinear, this is one of the most crucial property of such networks.

Input-output function. The overall ANN may be intended as a black box with a transfer function able to map input to output. Well performed training modifies synaptic weights in such a way that the transfer function behaves well. Being training algorithms automatic once provided input and target output (see Sec. 2.1.2 for deeper analysis on ANN training), user *does need not know* the transfer function itself, but just to train the net. Such a characteristic allows to model very complex systems or to spot features difficult to catch, without the needs of a strict mathematical and/or physical formalization of the system under analysis.

Robustness and adaptivity. Neurons' weights can change in time during operations to adapt to the environment, as in Adaptive Resonance Theory (ART). In this case, the trade is between *stability* and *plasticity* [52]. Moreover, the intrinsic robustness of the networks, deriving from their generalization feature, yields a great noise rejection capability and they can thus deal with partial and/or corrupted input data providing a meaningful output anyway. Even more, robustness of the net can be also exploited, in case of an hardware implementation, with a neuron failure. Performances will be certainly be degraded, but the neural system is still capable to manage the input output relation. It is possible also to implement a modification in the training algorithm to have better robustness [53].

Computational efficiency on parallel structures. Due to its highly

parallel structure, the neural networks well fits a parallel computing architecture. Nowadays, tendency to increase number of cores and CPU promises even better performances in the future for the neural networks. In particular, GPU with thousands of computing cores and very fast memories are definitely well suited for fast trainings.

2.1.2 Training

There are three main families of learning for the various types of neural networks:

Supervised. The training data is supplied with *desired* output vector (also called *target* vector). Scope of the training is to make the learner able to produce the correct output when a new input is presented [54, 55]. The present thesis deals mainly with this kind of machine learning.

Reinforced. A learner reacts to environmental stimuli producing an output. Acting on the environment itself, the learner is rewarded or punished depending on the goodness of its action. Aim is to be able to maximize the number of rewards (or vice versa for punishments). This kind of learning is particularly used in control systems theory [54, 55].

Unsupervised. The learner receives input data, but no training targets (as in supervised) nor rewards (as in reinforced) are given. The goal of such training is to find recurrent patterns in the input data. It is particularly useful for data mining of large datasets, in order to reduce their dimensions, without reducing the space of informations contained at the beginning [54, 56].

Specifically for this thesis, *supervised learning* has been employed to train multilayer ANNs, while *unsupervised learning* for SOMs (see Sec. 2.2). It is possible to identify two signal directions in a feed-forward network [44]: a *function (or input)* signal, that is the input signal propagating rightward (Fig. 2.4) layer by layer to the output; an *error* signal, originated at the output layer and back-propagating leftward towards the input layer. This last signal will drive the prince of the supervised training algorithms: the *back-propagation algorithm* that is a gradient descent based one. Among supervised learning

methods, the most common are *batch* learning and *online (or stochastic)* learning. They differ on how the error between the target and the actual output is computed.
Consider a training sample $\mathcal{T}$:

$$\mathcal{T} = \{\mathbf{x}(n), \mathbf{t}(n)\}_{n=1}^{N} \tag{2.4}$$

where $\mathbf{x}(n)$ represents the vector of N inputs, $\mathbf{t}(n)$ the vector of N targets. N is the dimension of the overall training set. Indicating with $y_j(n)$ the output of the neuron j in the output layer by the stimulus $\mathbf{x}(n)$, it is possible to define the j-th neuron's *error signal* as:

$$e_j(n) = t_j(n) - y_j(n) \tag{2.5}$$

That for all the neurons in the output becomes the *total error* in quadratic form:

$$\mathcal{E}_{|}(n) = \frac{1}{2}\sum_{j \in C} e_j^2(n) \tag{2.6}$$

where C contains all the neurons in the output layer.
Taking into account the overall training set, constituted by N examples, it is possible to define also the *mean error energy over the training set* as:

$$\mathcal{E}_{av} = \frac{1}{N}\sum_{n=1}^{N} \mathcal{E}(n) \tag{2.7}$$

$$= \frac{1}{2N}\sum_{n=1}^{N}\sum_{j \in C} e_j^2(n) \tag{2.8}$$

that is function of the synaptic weights, though not explicitly shown. As stated, batch and stochastic learning differ of the error taken into account to update neurons' weights. In **batch** learning, they are adjusted after the presentation of all the samples in $\mathcal{T}$, hence a suitable cost function to minimize though training can be $\mathcal{E}_{av}$. In **stochastic** learning instead, correction of the neurons' free parameters occurs sample by sample: each time a pair $\{\mathbf{x}(n), \mathbf{t}(n)\}$ is presented, weights are updated taking into account the error of that specific sample only. It is possible to express some considerations about the two methods:

- *Batch* learning involves a gradient descend that takes into account all the samples, and thus can be shown that its convergence to a minimum is guaranteed [44]. Note that convergence

is in general towards a *local* minimum of the error function. *Online* learning instead, because the gradient descent direction depends on the sample analyzed, is less subjected to be trapped in local minima.

- *Batch* requires much more memory to run, but can be easily parallelized. On the other hand, *stochastic* version is easier to implement and requires less memory, but it is composed intrinsically by serial operations.

In any case, the **back-propagation algorithm** is common to both the learning types.

The back-propagation algorithm.

The algorithm is the same for both batch and online learning, with the difference in the error computation. In the following, the error function to be minimized will be written as $\mathcal{E}$ and in case of batch learning one should refer to Eq. 2.7. There are two phases distinguishable:

Forward phase. Synaptic weights are unaltered, input signal is propagated towards the output layer and error $e_j = (t_j - y_j)$ is computed. j represents the output neuron index.

Backward phase. Error signal is *back-propagated* from the output layer towards the input neuron by neuron. For every neuron it is computed the local gradient, and weights are updated according the *Delta rule* (Eq. 2.16).

It has been already defined the net output (Eq. 2.2). Output signal of a neuron j is:

$$y_j(n) = \phi(v_j(n)) \tag{2.9}$$

The core of the algorithm is to apply a correction $\Delta w_{ji}(n)$ to the synaptic weight $w_{ji}(n)$. Such a correction is proportional to the error direction of variation. Through the chain rule it is possible to make explicit such a derivative:

$$\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = \frac{\partial \mathcal{E}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{2.10}$$

Analyzing each term of the chain, through Equations (2.6, 2.5, 2.9, 2.2):

$$\frac{\partial \mathcal{E}(n)}{\partial e_j(n)} = e_j \tag{2.11}$$

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \tag{2.12}$$

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \phi_j'(v_j(n)) \tag{2.13}$$

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n) \tag{2.14}$$

that yields

$$\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = -e_j \phi_j'(v_j(n)) y_i(n) \tag{2.15}$$

Note that with $'$ operator is intended the derivative with respect to the value inside parentheses.
Now, the rule to update weights is:

$$\Delta w_{ji} = -\eta \frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} \tag{2.16}$$

where η is the *learning-rate parameter*. Equation 2.16 is known in literature as **Delta rule**. The minus sign is requested in order to perform a gradient *descent* towards the (local in general) minimum.
Defining the *local gradient* $\delta_j(n) = \partial \mathcal{E}(n)/\partial v_j(n)$, through partial derivatives in Eqs. 2.11 ,the delta rule can be rearranged as:

$$\Delta w_{ji} = \eta \delta_j(n) y_i(n) \tag{2.17}$$

Because error signal $e_j(n)$ expression depends on the layer in which the neuron is, its expression has to be computed accordingly.
If neuron j is part of the output layer, expression is straightforward: desired response is directly known through the error signal itself in Eq. 2.5. Hence, local gradient and delta rule are provided:

$$\delta_j(n) = e_j \phi_j'(v_j(n)) \tag{2.18}$$

$$\Delta w_{ji} = \eta \delta_j(n) y_j(n) \tag{2.19}$$

For what concerns hidden layer neurons, desired response is not so straightforward: backward directed error signal starting from the output layer toward input must be taken into account. Exploiting chain

rule once again, local gradient can be redefined for a hidden layer neuron as:

$$\delta_j(n) = -\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \tag{2.20}$$

$$= -\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} \phi_j'(v_j(n)) \tag{2.21}$$

where j neuron belongs to hidden layer.
To compute partial derivative $\partial \mathcal{E}(n)/\partial y_j(n)$ it is necessary to go back to Eq. 2.6. Identifying with subscript k a neuron in output layer, whose neurons belongs to space C, the quadratic form of the error can be written as:

$$\mathcal{E}(n) = \frac{1}{2} \sum_{k \in C} e_k^2(n) \tag{2.22}$$

it is possible to compute the unknown partial derivative in Eq. 2.20:

$$\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial y_j(n)} \tag{2.23}$$

Exploiting again the chain rule allows us to write:

$$\frac{\partial e_k(n)}{\partial y_j(n)} = \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)} \tag{2.24}$$

and reminding Eq. 2.5, rewritable as: $e_k(n) = t_k(n) - \phi_k(v_k(n))$ and that the net output of neuron k is $v_k(n) = \sum_{j=0}^{m} w_{kj(n)_j(n)}$, where m is the total number of inputs:

$$\frac{\partial e_k(n)}{\partial y_j(n)} = -\phi_k'(v_k(n)) w_{kj}(n) \tag{2.25}$$

Therefore Eq. 2.23 becomes:

$$\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} = -\sum_k e_k \phi_k'(v_k(n)) w_{kj}(n) \tag{2.26}$$

$$= -\sum_k \delta_k(n) w_{kj}(n) \tag{2.27}$$

Through Eq. 2.26 and Eq. 2.20 it is finally possible to express the local gradient for a hidden layer neuron as:

$$\delta_j(k) = -\phi_k' \sum_k \delta_k(n) w_{kj}(n) \tag{2.28}$$

Note that activation function ϕ must be differentiable (see Appendix A for the most used activation functions).
One may notice that the back-propagation algorithm approximates the steepest descent to update the neuron weights. The *learning rate parameter*, can be tuned to provide a more aggressive algorithm, if increased, or vice versa. Care must be put in selection: a too high value could yield an unstable descent, while a too low one makes the training slow [51, 57, 44]. A part from using advanced algorithms with adaptive learning rate, a simple method to increase the learning rate avoiding instability is to include a *momentum* term in the delta rule equation:

$$\Delta w_{ij}(n) = \alpha \Delta w_{ji}(n-1) + \eta \delta_j(n) y_i(n) \tag{2.29}$$

where α is the *momentum constant*. This expression is usually called in literature *generalized delta rule* [44].

Other training algorithms

In this thesis, many training algorithms has been used to investigate their performances. Training time can be significantly reduced selecting a different method. This paragraph has the only intention to make the reader conscious about the existence of more refined methods, compared to the classical gradient descent. For technical details of the algorithms, it is possible to refer to the bibliography.

- *Resilient Backpropagation (Rprop)*. A very efficient first order batch optimization algorithm, it is one of the fastest available. It features an independent learning rate for each weight of the network. Weights are updated taking into account the *sign* of the gradient, not its magnitude [58]. The algorithm is divided in two parts: updating individual learning rates and update the weights. A pseudocode would look like:

```
while epochs<epochs_max:
  compute new error gradient;
  for each weight:
      if previous and current error gradient has same sign:
```

```
            increase delta;
            update weight with delta;
        else
            decrease delta;
            update weight with delta;
        end if
    end for
    epochs++ ;
end while
```

where "delta" represents the magnitude of the weight update parameter, responsible of the magnitude by which a neuron weight has to change.
Christian Igel and Michael Hüsken proposed an even faster version of the algorithm [59].

- *Gauss – Newton method.* Improvement of the classical Newton optimization method that involves computation of second order derivatives of the total error function to create the Hessian matrix [60]. In G–N algorithm instead, Jacobian matrix is calculated to approximate Hessian, therefore only first derivatives of the error function with respect to the weights. The training process is:

```
while epochs<epochs_max:
    compute Jacobian matrix;
    compute approximated Hessian;
    for each weight:
        update weights;
    end for
    epoch++;
end while
```

where weights update involves the approximate Hessian $H = J^T J$ as stated in [60]. Problems of convergence may arise if product $J^T J$ is not invertible [60].

- *Levenberg − Marquardt.* This method can be intended as the combination of a steepest descent pure gradient algorithm plus

the Gauss Newton implementation [61]. It solves the non invertible eventuality introducing a term in the Hessian approximation: $\mathbf{H} = \mathbf{J^T J} + \mu\mathbf{I}$ where μ is the *combination coefficient* and $\mathbf{I}$ is *identity matrix*. When the combination coefficient is almost null, the L−M methods is exactly the same as the Gauss Newton one. For high values of μ instead, the method approaches the pure first order gradient descent [60, 55, 44]. Pseudocode is the same as in G−N method, apart from the computation of the Hessian.

- *Scale Conjugate Gradient (SCG)*. This is a second order algorithm that exploits the conjugate gradient optimization technique. One of the main advantages of the SCG training algorithm is the low usage of memory, requiring only $o(N)$, where N is number of the weights in the network [62]. Moreover, it is particularly suited for unexperienced users, because it is fully automated. No manual selection of learning rate, momentum constant by the user. The scaled conjugate gradient results one order of magnitude faster than standard back-propagation [62].

Stopping criteria

The main parameter to take into account is to achieve a good level of generalization capability, that means to avoid *over-fitting* or *under-fitting*. The first refers to the problem occurring when the network is "too trained" and memorize the data without understanding the underlying function or pattern between the various inputs, that is, for example, to learn the noise inside the training set. Under-fitting is the opposite: whenever a network is unable to generalize because of the excessive complexity of the set. To overcome the last, usually it is sufficient to increase the synapses of the network, adding layers and neurons in general, or to pre-process the input to better distinct the various features hidden in the model.
There are several options to stop a training: *maximum number of epochs* and *training time* are usually exploited to prevent slow o non-converging trainings to keep on going indefinitely. None of them takes into account the effective "goodness" of the trained net. Statistical *regularization* techniques may be adopted to prevent the weights to assume extreme values [63], thus they work limiting the weights

space inserting or decreasing the probability of more complex model (i.e. Bayesian learning [55, 63, 64]).
The most used approach is *cross-validation* [44]: the data set is divided into three subsets:

- *Training subset*, that represents the data actually subjected to the training algorithm and that will forge the model the training network will approximate;
- *Validation subset*, used to validate and to stop the training when a certain constraint is true;
- *Test subset*, to assess the performances reached by the artificial neural network through the training.

Typical proportions between subsets are in the order (60, 20, 20) respectively for training, validation, test. The motivation of this approach is to validate the model on data different from the one used to compute neural network parameters in training.
Normally, through training algorithms, the mean squared error of the output decreases epoch by epoch. It starts large with weights just initialized randomly, decreases rapidly and then slows down approaching the local minimum of the surface error. To end the training obtaining a good generalization capability, looking at the training error is not enough. Through the validation set the network is periodically tested during the training and mean squared error (or other performance parameter) is evaluated: if the validation subset error is at a local minimum (that is when for N consecutive validation subset error increases), training is stopped to prevent *over-fitting* [44, 51, 57].

2.1.3 Empirical notes on how to train effectively

Due to the fact that for large datasets and complex network architectures, training may be time and resource consuming activity, a short series of advices mainly reported from personal experience and [44, 51, 65] to train fast and effectively is here presented.

- *Online and batch learning.* If large parallelized computing frames are available, it is usually better to use batch training, because of the large number of optimization techniques available. Otherwise, stick with the simpler online version, that saves memory too.

- *Selecting the dataset.* The importance to choose the right samples in dataset is fundamental: the most various examples will be given to the network during training, the better performances will be achieved during operations. Avoiding similar data prevent the training to be unnecessary over-headed without improving performances in generalization.

- *Normalization and transformation of the inputs.* It is usually a good norm to *normalize* all the inputs to have average value about zero, so to not incur in the case where all the inputs are of the same sign and thus to update the weights of the first layer with the same sign (proportional to the local gradient). This provokes the weights to increase or decrease at each iteration all together, with consequent slow convergence. Another good practice is to *scale inputs* such that their covariances are almost the same: this yields the same importance to all the inputs present in the dataset. On the other hand, if some inputs are less important than others, their values should be scaled down. Third good practice should be to *decorrelate* inputs, in order to have a diagonal like system of equation in which one input does not depends on the other.

- *Symmetric sigmoids.* Especially for hidden layers neurons, use symmetric sigmoids as activation functions because they yield faster convergence due to the fact that their output -that is the input of the next layer- is more likely with average zero. That gives a higher first derivative magnitude, that eventually drive the delta rule to be larger[4]. Care must be posed to prevent *overflow* when the input to the sigmoid function is very large in modulus, for example assigning -1 or 1 for every input $x \geq |\beta|$ where β is a convenient constant.

- *Initialization of the weights.* A good initialization of the weights can tremendously reduce the amount of training time. There exist both empirical and theoretical derived approaches. See [44, 51, 65]

- *Learning rate and momentum selection.* Choosing a proper value for the learning rate yields much faster convergences. It is more

[4]Not true for algorithms like the Rprop because their weights updates are not proportional to the magnitude of the gradient

convenient to use different learning rates for each weights, in order to optimize each neuron. Ideally, the learning rate value should decrease when the error starts oscillating and increase otherwise. Many algorithms to adapt individually learning rate of every neuron exist in literature [65]. Adding the momentum term increases stability of the algorithm.

2.2 Other types of Neural Networks

Another kind of artificial neural network architectures is now presented: they have been used in this research study and they are also planned to be investigated in future works to affine the hazard detection algorithm.

2.2.1 Self-Organizing Maps

Self-Organizing Maps (SOM) differ from the previously debated feedforward networks due to the fact that the correct output is not known *a priori*. Therefore, no performance quantity -like MSE- used to measure the behavior of a supervised trained network can be used.
Each time an input vector, representing the *environment*, is presented to the SOM, the network's internal parameter are changed. If such changes are correctly driven, the SOM is able to map the space into its output space, that depends on the dimension chosen for the output layer of the network [51]. Hence, the SOM is able to create a representation of the environment. The most well-known network of this kind is the Kohonen's [66].
These maps work through a topological principle, inspired by visual cortex of the brain in mammals, that is able to recognize three dimensional objects even though it is a bi-dimensional (planar) structure. Kohonen maps are relevant in the definition of the neighborhood, that is to create the correct topological structure during the learning, to transform the input in a n-dimensional output space (typically bi-dimensional) and thus lowering the space dimensions without loosing informations.
For bi-dimensional case, a Kohonen net features a single *computational/output* layer fully feedforward connected with the input layer. The output layer is composed by a lattice of neurons. The objective of the

learning process is to make a topological map in which every region of the lattice is specialized to map certain input regions [66].

Learning process algorithm for Kohonen nets.

Let be n the dimension of the input vectors, m the number of neurons in the computational layer. For every computational unit, it is possible to define a radius r that includes the corresponding units in the neighborhood of the current unit. Each neuron of the lattice computes the euclidean distance between the input $\mathbf{x}$ and its weight vector $\mathbf{w}$. A neighborhood function ϕ computes during the training the relation between the current unit and units inside or outside to the radius r.
After initialization of the n-dimensional network weights and selection of the initial radius r, initial learning rate η, neighborhood function ϕ, the algorithm develops in such a way:

```
while epochs < epochs_max
    -Selection of current input vector ξ;
    -Compute unit k with min distance;
    -Weights vectors of the unit are updated:
     w_i = w_i + ηφ(i,k)(ξ − w_i) for i = 1 : m;
    -epochs++;
```

The update of the weights attracts the neurons of the neighbourhood towards the input ξ. With a sufficient number of iterations and a complete enough input space, also the computational units' weights should distribute uniformly. If learning process is well performed, the resulting Kohonen net should be able to statistically quantify correctly the input space in their discrete output space [51, 44].

Chapter 3

ORACLE+ hazard detection system

In the following sections, a description of the ORACLE+ hazard system architecture, developed at Politecnico di Milano by Alessio E. Colombo [67] and inspired by Simone Bernardi's ORACLE system [68], will be presented to the reader. To understand the reason that encouraged further development in the present thesis, it will be also depicted in Sec. 3.2 the main problems that affected the previous work.

3.1 Architecture

The Hazard Detection System designed in [67] is specifically tailored for lunar landings. It is designed to process images from a single camera and to provide to the system a landing site. It is composed by (Fig. 3.1):

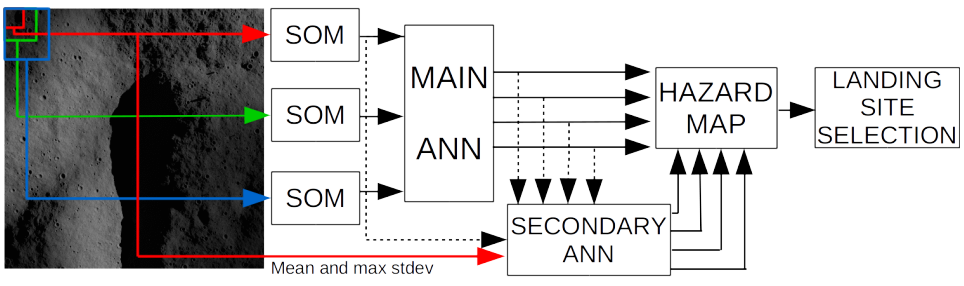


Figure 3.1: ORACLE+ system architecture.

- Image preprocessing and indices extraction algorithms;
- 3 Self Organizing Networks that cluster indices extracted from the image before actual processing by feedforward neural networks;
- 2 Feedforward multilayer artificial neural networks: the Main one, devoted to the computation of the hazard index, the Secondary one, to improve the capability of the Main one, in particular to recognize slopes;
- Creation of the hazard map through the output of the feedforward neural networks;
- Landing site selection algorithm.

3.1.1 Image loading and indices extraction

The image acquired by the camera is firstly converted in 8 bit gray scale. Then 3 mobile windows of dimension respectively 8×8, 16×16, 32×32 extract the indices to be classified through the Kohonen maps: *mean* μ and *standard deviation* σ:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} I_i\,, \qquad \sigma = \sqrt{\frac{\sum_{i=1}^{N}(I_i-\mu)^2}{N-1}} \tag{3.1}$$

where N represents the number of pixel considered for the mean, I_i the intensity of the i-th pixel.
Indices are computed in four directions for every window size as depicted in Fig. 3.2. Therefore for every window a vector of 8 elements is obtained. Values are normalized to prevent neurons saturation, with domains transformed to [0,5] and [0,10] for mean and standard deviation respectively.

3.1.2 SOM networks

Three Kohonen nets, one per window, are exploited in order to reduce the input space for the Main feedforward multilayer neural network. Computational layers dimensions for the SOMs have been selected in 10×10 neurons. Thus, the three nets are able to categorize the input into 100 classes.

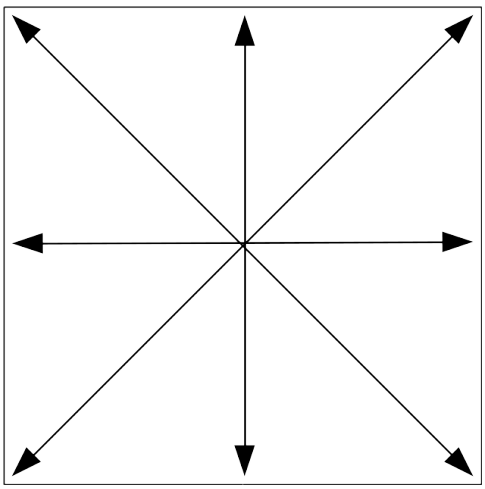


Figure 3.2: Directions used to compute mean and standard deviation in a single mobile window.

3.1.3 Main Feedforward ANN

Outputs of the three SOMs are assembled to form the overall input for the Main feedforward network. There are 4 indices to assess hazardousness of a particular surface region:

- *Safe*. A planar region without craters, rocks, shadows and slopes.
- *Slightly safe*. Like safe, but with a certain surface roughness.
- *Unsafe*. Highly rough area, with eventual fractures.
- *Highly unsafe*. Regions with dangerous morphological structures, such as crater rims, harsh surfaces, high slopes and shadows.

3.1.4 Secondary Feedforward ANN

A second Feedforward network has been added in order to better classify slopes and hazard index in general: it is fed with outputs of the Main ANN, the mean across overall the pixels of the small window and the maximum standard deviation value encountered in the 4 directions for any single window. A slope is typically bounded by an edge, that is a strong variation in pixel intensity, thus it is easy

identifiable with standard deviation. Thus, combining its results with the Main feedforward network, better classifications are possible.

3.1.5 Hazard map computation

Output of the two feedforward networks is combined to provide one hazard value for every small window, so that an hazard map side will be divided by eight with respect to the original image. The actual output from the Main ANN is a four elements vector, in which each component belongs to domain [0,1]. Ideally, output should be an integer 1 corresponding to the hazard index of the region analyzed, 0 for the remaining elements. Actually, if training has been well performed, output will be elements approaching 1 and the others almost 0 for a correct identification of the hazardousness. Secondary ANN, instead, yields a 4 elements output vector with values' domain of [1,4], assigning for each window an additional scalar parameter. Such a coefficient can modify the hazard index of the Main multilayer ANN in order to increase accuracy and flexibility. The Hazard Index (HI)[1] for the window i is computed:

$$HI_i = c_{i,1}o_{i,1} + c_{i,2}o_{i,2} + c_{i,3}o_{i,3} + c_{i,4}o_{i,4} \tag{3.2}$$

where $o_{i,j}$ is output element j of window i from the Main multilayer network, $c_{i,j}$ is output element j of window i from Secondary multilayer network.

3.1.6 Landing Site selection

To select the landing site, the hazard map is analyzed dividing it in a series of different hazard levels and selecting in each of them the most suitable landing site. The lack of a ground truth coerces to not impose *a priori* any maximum or minimum threshold, but every level of the hazard map is shifted downward of a quantity corresponding to the mean between minimum and maximum of the hazard indices found in the image. Hence, every negative value is considered safe, whereas a positive level will be assigned to unsafe levels. To calculate the *best landing site*, first of all, area and centroid of the safe regions are circumscribed. Areas that are too small with respect to the lander

[1] Original italian nomenclature: "Livello di Pericolosità (LDP)"

footprint are immediately discarded. Then, the landing site ellipse is computed on the remaining regions:

- distance of hazard pixels from the centroid are computed: starting from the centroid itself, towards eight directions, pixel by pixel the program approaches the first hazardous spot;
- the centroid is shifted in a more safe position: depending on the polygonal shape of the safe region, centroid could also be nearby a dangerous point;
- an ellipse centered in the new centroid is computed. Its semiaxes are enlarged up to the pixel before an unsafe point.

After the computation of all the landing site candidates, it is necessary to select the best one for every level. Therefore for each ellipse it is computed the *distance functional*:

$$F = \sqrt{A^2 + d_{obstacles}^2 + d_{borders}^2} \tag{3.3}$$

where A represents the landing site area, $d_{obstacles}$ and $d_{borders}$ respectively the distance between the landing site candidate and the nearest obstacle and the distance between the candidate and the image borders.
Site that maximizes functional F will be chosen as candidate for that level. After this phase a vector containing at most as many elements as the number of levels are available. To choose among those, a second functional is exploited:

$$F_{final} = \sqrt{F^2 + level^2} \tag{3.4}$$

where F is the functional computed in Eq. 3.3, $level$ the hazard level of the corresponding functional F. Therefore maximization of F_{final} takes into account also the hazard index.

3.1.7 Training set creation

In a first design iteration, artificial lunar surface images have been used to perform training, but they have been discarded soon afterwards because they were judged unrealistic [67]. Hence, real lunar images have been used to train the neural networks in the hazard detection system. In order to create an automatic training system,

without the need of manually specify the hazardousness of a region, it has been implemented a *fuzzy logic* based interpreter, whose architecture is shown in Fig. 3.3.

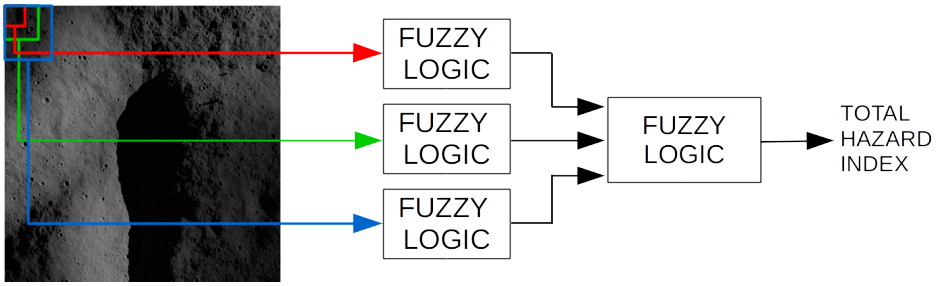
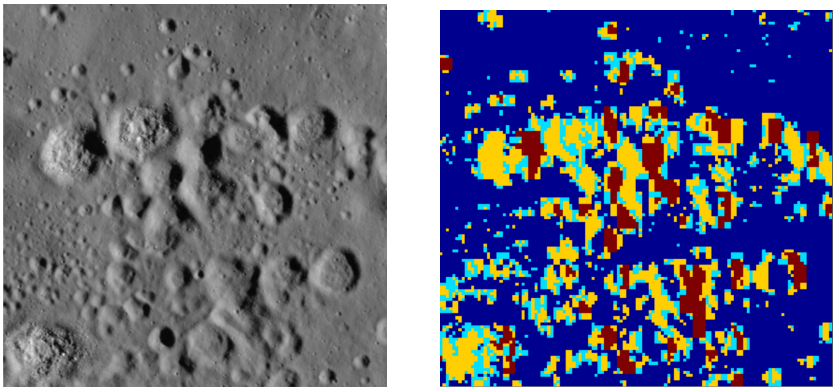


Figure 3.3: Training set creator, ORACLE+.

Also in this case, three windows spanning the whole training image are present. Windows dimensions have been kept the same 8×8, 16×16, 32×32 px as in the hazard detector. Per each window, mean μ and standard deviation σ of the whole pixels distribution are computed, and a fuzzy logic block extrapolates the hazard index. An additional fuzzy logic block combines the three single hazard indices to give a single value. To have a precise description of the system and the fuzzy logic rules adopted, see [67].
An example of hazard map through the fuzzy logic based system is given in Fig. 3.4.



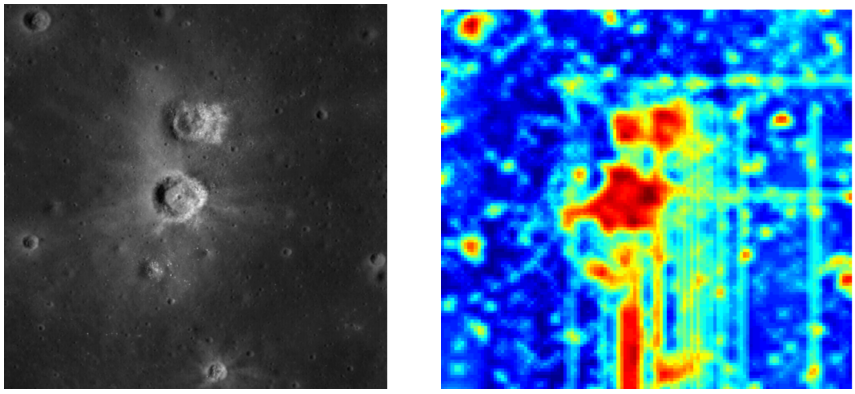
(a) Camera image (b) Hazard map

Figure 3.4: Lunar image used for training and corresponding hazard map computed with fuzzy logic.

3.1.8 Results

Some of the results by A.E. Colombo are presented to have an idea of what the old hazard detection system was capable of. Following test images are not included in the training set, thus are completely unknown by the hazard neural systems. Note that all the test images hereby depicted are real lunar photographs. A first example is represented in Fig. 3.5: a planar area dominated by a low albedo, with a couple of dangerous craters in the center and some smaller ones randomly distributed. Flat terrain seems correctly cataloged as safe, while the various craters and nearby areas are considered unsafe. On the other hand, large part in center bottom image appear dangerous, although they appear as safe as the other planar regions in the picture.

Another test image is presented in Fig. 3.6. Particularly critical in

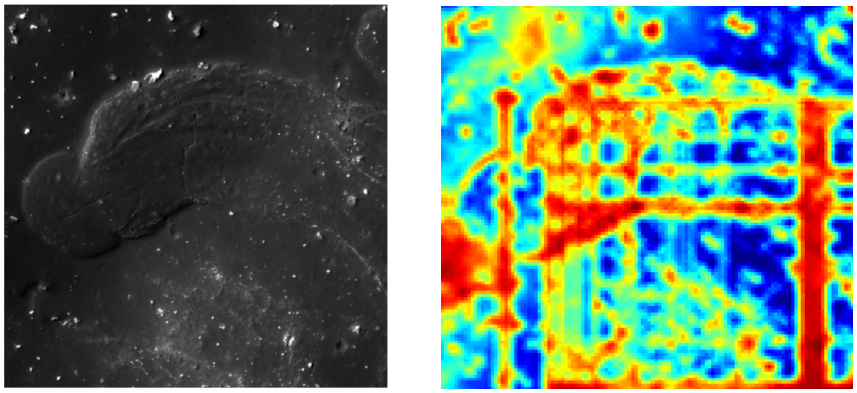


(a) Camera image (b) Hazard map

Figure 3.5: Test image. Planar region with craters and different soil albedos.

this case is the capability of the hazard detection system to understand the long edges and the different heights of the features present in the picture. The central plateau's edges are almost recognized, but hazard indices looks like not always coherent with the roughness or the apparent slope of the image, especially planar areas that are cataloged as dangerous. Nevertheless, the general structure of the image looks recognized by the networks: edges of the plateau is almost everywhere recognized, such as the majority of small craters present.

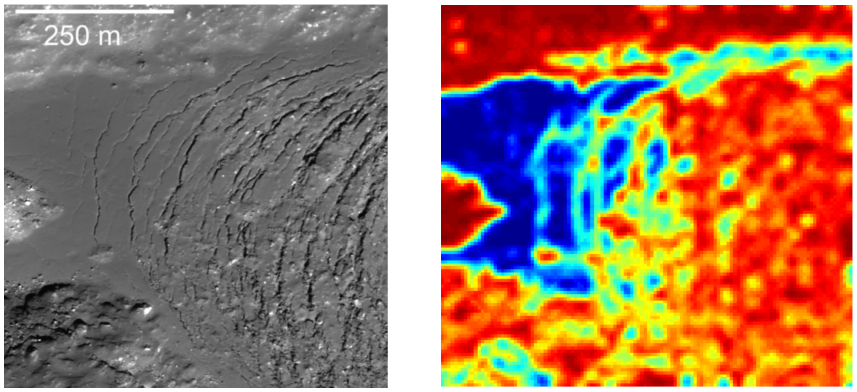
In Figure 3.7 a picture of the crater *Moore F* floor is shown as test image. It is characterized by numerous fractures and a very irregular



(a) Camera image (b) Hazard map

Figure 3.6: Test image. Dark region with different slopes and long edges.

terrain, a part from small areas that looks like the most fitting landing site candidates. In the relative hazard map, the system catalogs the upper area as the most dangerous, probably misled by the change in albedo. Fractures are recognized correctly as a dangerous area, but some small regions in between should have been output as safe, or at least not completely unsafe. On the other hand, the big planar area in the left hand side of the picture is coherently computed as safe.



(a) Camera image (b) Hazard map

Figure 3.7: Test image. Floor of the crater "Moore F".

3.2 Limitations

In Section 3.1.8 a *qualitative* overview has been proposed for the hazard map computation. It has been noted that the hazard maps tends to understand correctly part of the features of the various test images, even if they are characterized by imprecisions and sometimes wrong measurements of the hazard index. Anyway, the main problem of the hazard detection architecture is the *completely non objective* ground truth. In fact, the system described in Sec. 3.1.7 cannot be said to yield a *true* ground truth: no morphological values of the lunar surface have been used to compute hazard index of the images. Therefore, even a perfectly resulting hazard map computed by neural network cannot be said to be good or not, due to the fact that no validations can be made because of the lack of a *true* hazard map to compare the computed map with. Moreover, the output of the main ANN –trained with the dataset created with a fuzzy-logic interpreter– was corrected through the secondary ANN manually assigning the target that the HD system should have provided, as guessed by the user. Hence, ORACLE+ system is constrained to the operator's subjective judgment about the hazardousness of the various lunar surfaces, making this systems results not reliable.

Chapter 4

Training, validation and test datasets

As it has been pointed out in Chapter 3, the robustness of a hazard detection system can be assessed only by an objective validation process. Hence, it has been developed a ground truth generation program that allows radically better training and performances assessment of the system through the use of high accuracy artificial images.

4.1 Ground Truth Hazard Maps from Digital Elevation Model

The choice to train and test the neural network through *artificial* images provides solid benefits with respect to the use of real images, where no actual ground truth is available:

- *An accurate and objective knowledge about safe and unsafe landing sites*. That is thanks to the fact that physical parameters, like slopes and roughness, are known in the artificial images through DEM and rendering data, and therefore they can be cross-checked with the lander characteristics to provide a measurement of the hazardousness of an area.
- *Multiple image with different parameters of the same area are possible*. Changing Sun inclination or lander attitude it is possible to expand the capability of the neural network, training it to deal with different situations. Those two values are always available

on-board a spacecraft, but almost never included in real images metadata.

On the other hand, artificial images needs to be *realistic* enough to provide a *coherent* dataset to the neural network. Otherwise, the system may be not useful at all when tested on real lunar images, due to training set incoherence. Thus, it has been exploited one of the richest and most accurate DEM data available, that is the Lunar Reconnaissance Orbiter Camera (LROC) set [69]. The available resolution of the complete lunar surface is 100 m/px, with several subregions released at a higher detail between 2 and 5 m/px. Such value is anyway too low for hazard detection and in general navigation during the last phases of the descent. Therefore, small scale detail has been added by inclusion of fractal noise, craters and boulders [70, 71, 72]. Fractal noise has been added through the diamond-square algorithm [73] that modifies the altitude of points between the DEM data randomly in the values range of the segment length, yielding a realistic effect. It should be noted that a simple interpolation of the DEM original points would surely smooth the terrain surface but would increase only *virtually* the resolution, without adding any detail to the lunar soil, resulting therefore useless to increase the details of the lunar surface. Craters has been inserted artificially following the real statistical distribution present in the lunar surface, while their morphology is built taking into account empirical data deriving from lunar craters observations [74, 71]. Their largest size is equal to the maximum non reproducible by the original DEM resolution. Hence, only small scale craters are artificially added. Through all these improvements, resolution of the DEM has been improved to 0.3 m/px [70]. At this point, a photorealistic image is rendered trough ray-tracing techniques provided by the software POV-Ray [75], taking into account the camera parameter summarized in Table 4.1. This DEM image is colored with a grayscale texture with colors spacing between 0.3 (darkest) and 0.8 (brightest) out of 1. Color is selected according to the surface roughness: brighter where terrain is rougher and darker where smoother. Rendering is performed taking into account Sun light, radiosity and setting the ambient light source –to simulate inter-diffuse reflection– to 1%.

About each DEM point, a circular window of the dimensions of the lander's footprint selects the corresponding nearby points, and it is

Table 4.1: Parameters used for POV-Ray renderings.

Camera model	*pinhole*
Image resolution	*1024×1024*
Angle	*60°*
Color	*8 bit gray scale*

thus possible to compute *slopes* and *roughness* of that specific surface area, directly from DEM data.
Slope is quantified as the inclination of the mean plane composed by the circular window points, approximated through a least square approximation. Through the plane equation

$$z = ax + by + c \tag{4.1}$$

where x, y, z represents respectively the two window coordinates and the height, it is straightforward to obtain the slope S:

$$S = \arctan(\sqrt{a^2 + b^2}) \tag{4.2}$$

Roughness R is instead taken into account through the difference between the maximum and minimum deviation of the DEM window points i with respect to the mean plane:

$$R = \max_i\{z_i - (ax_i + by_i + c)\} - \min_i\{z_i - (ax_i + by_i + c)\} \tag{4.3}$$

Directly in POV-Ray, coordinates transformation from DEM to camera perspective is executed and it is possible to impose the safety constraints:

- $S \leq S_{max}$
- $R \leq R_{max}$
- The point is in light.

At this point, hazard map ground truth is calculated:

- Slope, roughness and shadows *boolean maps* are computed applying a threshold according with the safety constraints S_{max}, R_{max} and shadow presence. In these maps, 1 will be assigned in

unsafe value, 0 to *safe* ones. They represent the boolean textures that will generate three intermediate images necessary to create the ground truth hazard map. In particular, the rendering of the three boolean maps is computed with these parameters:

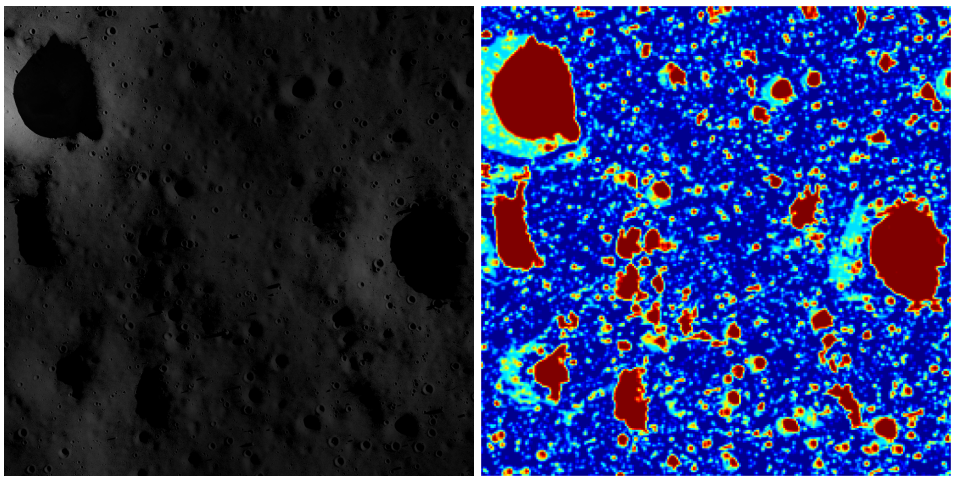
  - *Shadows map* rendering: no diffuse ambient lights nor their radiosity due to Sun light are rendered, only direct Sun light. White texture is used for terrain. A map composed of net shadows result.
  - *Slopes map* rendering: diffuse ambient light is computed, while direct Sun light is off. Texture map used is the boolean slopes map. This way, a map depicting only safe and unsafe slopes is obtained for the whole image.
  - *Roughness map* rendering: light parameters are the same as for the slopes rendering map, but texture map is the roughness boolean and therefore it results a picture with safe and unsafe points considering only roughness of the terrain.

- Through boolean renderings of slopes, roughness and shadows, hazard map HMAP is computed, having care to normalize to 1 the hazard index:

$$\text{HMAP} = \max_i \left\{ \frac{1}{3}(S + R), W \right\} \tag{4.4}$$

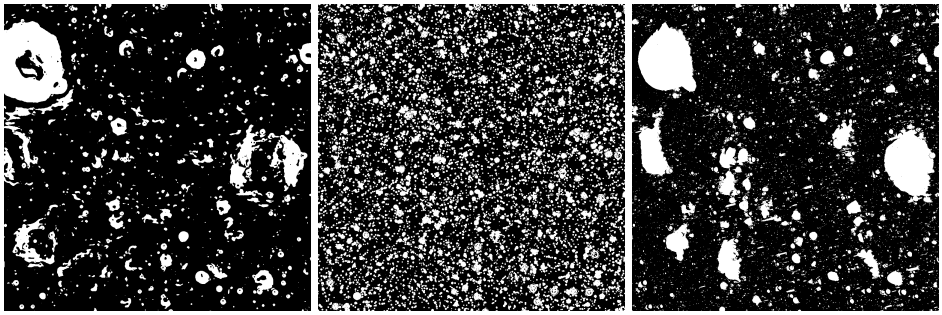
  where S, R, W are respectively the slopes, roughness, shadows boolean maps (Fig. 4.2).

An hazard index of 0 means a *safe* location, 1/3 *unsafe* with respect to roughness or slope, 2/3 *unsafe* for both roughness and slopes, 1 *unsafe* for being in shadow, that are areas out of the camera field of view. Thus it is an hazard value increasing from 0 –safest– to 1 –most unsafe–.
Eventually, two Gaussian pyramids downsample the image resolution down to match the output size of the neural network. As it will be properly explained in Chapter 5, downsample is necessary mainly due to computational performances reasons, that cannot be omitted since the whole hazard detection system must run real-time. Moreover, a downsample allows to relate nearby pixels, obtaining a much smoother ground truth hazard map. This increased continuity in pixels' hazard index theoretically facilitates the training process of



(a) DEM rendering (b) Hazard map

Figure 4.1: DEM photorealistic rendering and corresponding hazard map. Intermediate rendering maps that concur to create (b) are shown in Fig. 4.2



(a) Slopes map (b) Roughness map (c) Shadows map

Figure 4.2: Intermediate stage renderings for the hazard map computation.

the ANN, since it is much more effective approximating a continuous function with respect to a discontinuous one [70].

4.2 Training, Validation and Test datasets

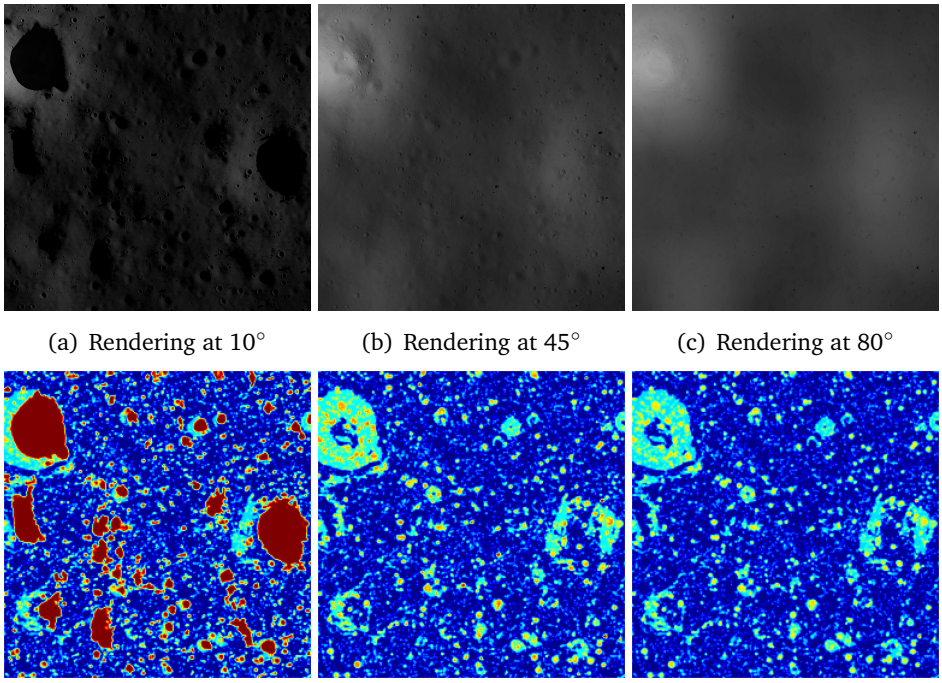
As introduced in Chapter 2, an adequate training set is fundamental to obtain a neural network capable to yield great generalization performances. The various features that are expected to be encountered during operational run-time should be included as completely as possible. To achieve such a goal, it has been generated *a set of 98 images*

and hazard map ground truth from 34 different DEMs. Such a dataset is composed by:

- *Training dataset*. 72 doublets[1] with Sun inclination angle of 10°, 45°, 80°. Therefore, 26 different DEMs rendered at 3 different Sun inclination angles has been used. So, the network will be trained with images characterized by net shadows, when the Sun inclination value is low, and with pictures where the various features are less sharp, when the Sun looks higher to the planetary surface.
- *Validation dataset*. 4 DEMs creating 12 doublets with Sun inclination angle of 10°, 45°, 80° have been exploited. As said in Chapter 2, this set is used to assess the error performances of the network while training in order to avoid over-fitting trough early stopping.
- *Test dataset*. 8 doublets generated with 4 DEMs at 15°and 80°. This set is exploited once the network has completed the training, to assess the performances of the overall hazard detection system. It is not used for training purposes. Test dataset is available to the reader in Appendix B.

DEMs used in each one of the three sets are different with respect to the ones used in another set, so that it is possible to have a performances evaluation of the hazard detection system on images totally uncorrelated with respect to the training.
As the reader may have noticed, almost the whole Sun inclination angle spectrum has been utilized to both train and test the network. This is a fundamental issue, since the hazard detection system proposed in this thesis is aimed to deliver a safe landing site with every light inclination. Azimuthal position of the Sun, instead, is fixed at 15°, since it does not enrich the training set coherence: hazard detector has not a preferred direction in the image frame plane.

[1]From now on, with doublets it will be intended a pair of lunar surface image rendering and its relative ground truth hazard map.



(a) Rendering at 10° (b) Rendering at 45° (c) Rendering at 80°

(d) Hazard Map, 10° (e) Hazard Map, 45° (f) Hazard Map, 80°

Figure 4.3: Renderings at 10°, 45°, 80° and respective ground truth hazard maps. These images have been used for the training dataset.

Chapter 5

Architecture of the Hazard Detection System

In this chapter the current architecture of the system will be presented. The most important alternatives that have been investigated during the design and the development of the hazard detection system will be briefly discussed.

With respect to the system described in Chapter 3, the current version is definitely simpler, as it can be seen in Figure 5.1:

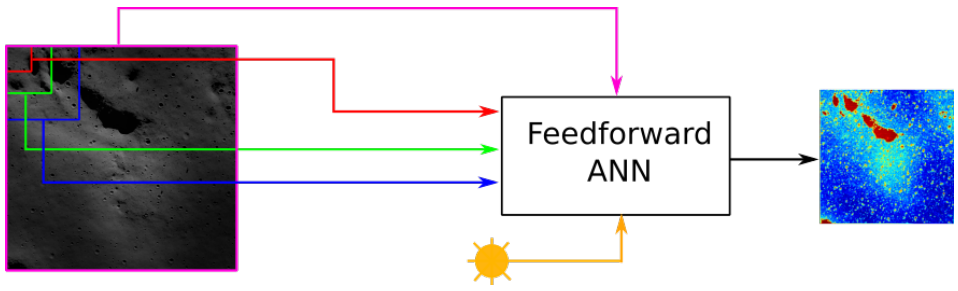


Figure 5.1: Hazard Detection system architecture.

- Image preprocessing and loading has remained mainly unchanged: 8 bit gray color depth is maintained. Only resolution of the input frame has been increased to 1024×1024;

- The three Self-Organizing Maps have been removed. The way they were implemented did not yield performances increasing: a clustering of the data coming from the image processing it is redundant, due to the fact that the image is already cataloged through its extracted indices;
- The secondary Feedforward Multilayer Artificial Neural Network have been removed: through a better image features mining, a larger and smarter training dataset with *true* ground truth hazard maps available, a single –and even smaller in terms of neurons and layers– Feedforward multilayer ANN is perfectly capable to operate.

Therefore it is possible to refer to a new system architecture, shown in Fig 5.1

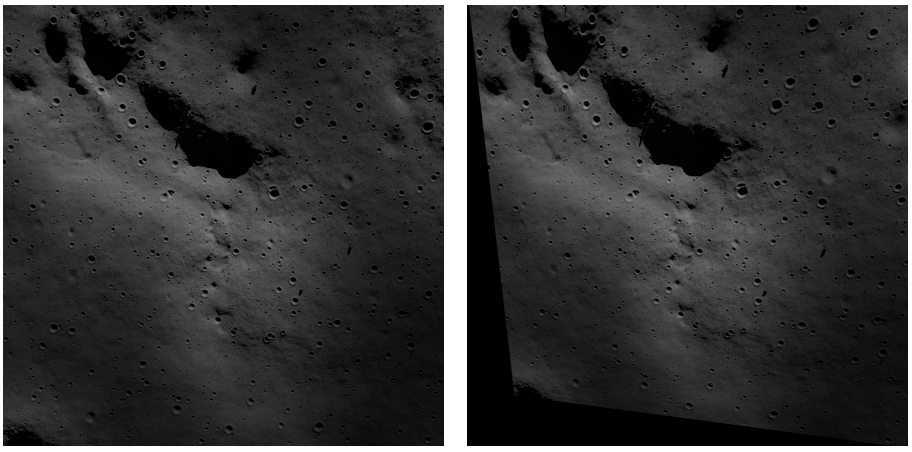
1. Image loading and preprocessing: the raw image is acquired. Image perspective correction is applied if needed.
2. Indexes extraction: image is segmented at different scales with various indices and low level informations are extracted.
3. image indices are processed by a feedforward back-propagation artificial neural network and arranged in the hazard map.
4. Target Landing Site search. Computed hazard map is exploited to select the most suiting landing site.

5.1 Input image preprocessing

Image is first of all loaded in 8 bit gray scale and if it is rectangular in shape, it is cropped to square size. Dimensions of the processed image are defined by the maximum centered square that is contained in the original rectangular image. Then, a resizing occurs if image width (or height, since it is now squared) size is greater than 1024 px, since the overall hazard detection software is programmed to work with 1024×1024 input images. Such a size have been selected because of the compatibility with many of space proven camera devices. Note that this preprocessing function is not needed if the images used are the artificial ones described in Section 4.1, since they are already

1024×1024 px in size. Instead, image crop and resizing will be exploited during the tests on real lunar images (Sec. 6.5), where their resolution is in general different. At this stage, if it is needed, another image processing routine is called: due to the fact that the system is designed to operate with images shot from a vertical attitude as the spacecraft is assumed to start the hazard detection phase in vertical attitude at the HDA High Gate point of the landing phase (Section 1.2.2 presents the whole maneuver in details), it is possible to correct any small deviation from the vertical attitude through a *perspective transformation* [70]. It may be the case when the lander is descending with a non completely vertical attitude. Such image transformation is computed under the hypotheses of planar surface. The knowledge of the altitude, the aperture of the camera Field of View (FoV) cone and the current attitude of the spacecraft, allows to compute the projection of the image points in real world reference. Then, such points are converted in the camera frame and it is possible to compute the *perspective transformation matrix* that allows to transform the original image into the perspective view [76, 77, 78]. An example of such image processing routine effect is depicted in Fig. 5.2.

Moreover, if the images are provided by a real camera, radial and



(a) Original image (b) Transformed image

Figure 5.2: Perspective Tranformation. FoV: 60°, altitude: 2000 m, pitch: 7.1°, yaw: 7.1°, roll: 0.1°

tangential distortions generated by the lens imperfections are corrected.

5.2 Indices extraction

Significant indices are extracted from the image in order to supply the following neural network with the most appropriate informations to achieve high performances maintaining an elevated computational efficiency. As the reader may grasp, many kinds of indices have been evaluated to provide the best results. The most significant discarded versions of the indices extraction algorithm are reported in Sec. 5.6. In general, the informations to extract from a lunar surface image, must be such that the network is able to detect the various planetary soil features that make a site unsuitable to land on, such as slopes, craters, scraps, as correctly as possible. To accomplish this task with a single camera, it has been exploited either *zeroth*, *first* and *second differential orders* of the gray scale image pixels intensity.
Zeroth order indices allow to have a reference value from which the network can understand the general brightness of the area analyzed in the acquired image. One the other hand, higher orders are dedicated to detect the various features present on planetary surface detecting variations -and how fast the variation occurs- of pixels intensity. Moreover, it has been adopted a multi-scale approach for the various indices: exploiting multiple scales of the image through the downsample of the original one, makes possible for the network to understand depth and relative distances of the objects and the features present in the image under analysis [79]. Specifically, the current version of the hazard detection system consists of two sets of indices:

- *Window-based* indices. Three different mobile windows are exploited to compute mean μ and standard deviation σ of the pixel intensities of partial image regions. Sizes of those windows have been chosen balancing computational performances and results accuracy in terms of landing sites found by the neural network. Three windows s_S, s_M, s_L of size respectively of 4×4, 8×8, 16×16 px have been selected.
  The two statistical indices used are defined as:

$$\mu = \frac{\sum_{i=1}^{N} I_i}{N} \tag{5.1}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N} (I_i - \mu)^2}{N-1}} \tag{5.2}$$

where I corresponds to the intensity of the i-th pixel, N is the number of pixels inside the considered image window. Mean helps the network to reconstruct the general intensity of the window considered, and it is particularly useful to distinguish shadows -or deep space- from the rest of the planetary surface. Standard deviation is exploited to yield a first assessment on the variations, with respect to the mean value, of the window intensity, providing the net a general trend of the features present in the current window.

- *Global* indices. Image gradient (grad) and Laplacian of Gaussian (LoG) are computed on the whole image. These two operators have been approximated through convolution with custom 5×5 linear kernels. The image is then downsampled to match the dimensions of the window-based indices, therefore down to 256×256, 128×128, 64×64. Those very dimensions allow to match the output size of the window-based indices processing. *Image gradient*, that represents the first order derivative filter, is implemented with two directional expanded 5×5 Prewitt [80] filters kernels, P_H and P_V that approximate the image gradient respectively in the horizontal and vertical directions [81]:

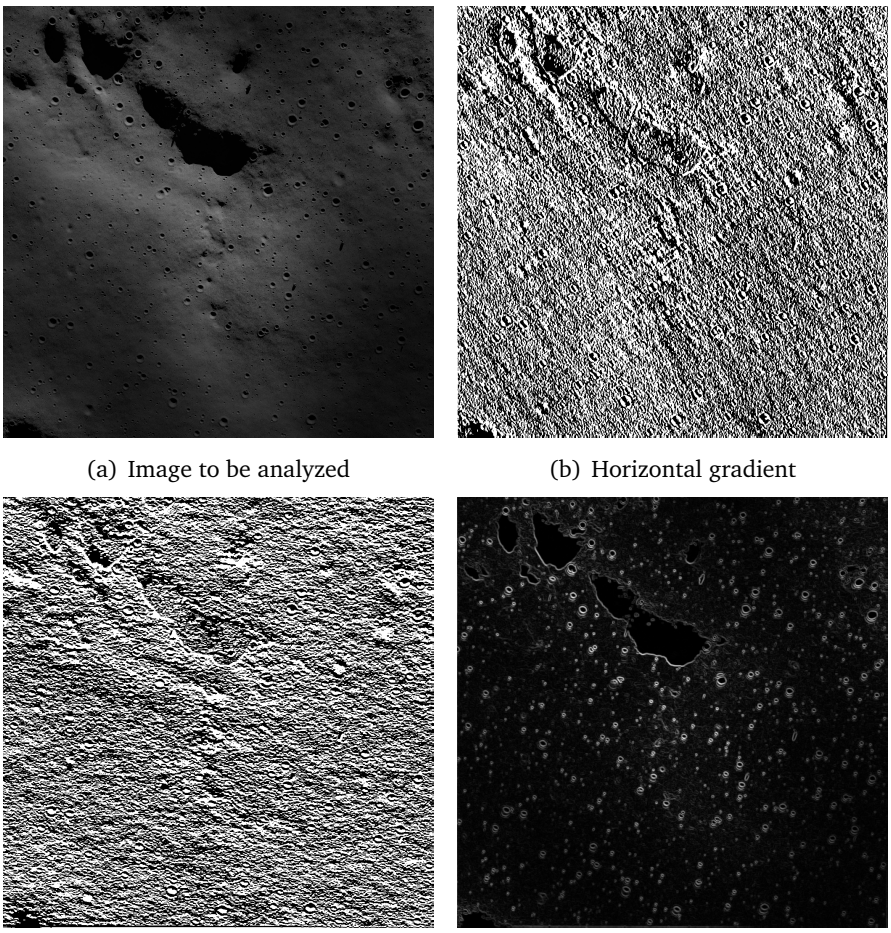
$$P_H = \begin{bmatrix} 2 & 1 & 0 & -1 & -2 \\ 2 & 1 & 0 & -1 & -2 \\ 2 & 1 & 0 & -1 & -2 \\ 2 & 1 & 0 & -1 & -2 \\ 2 & 1 & 0 & -1 & -2 \end{bmatrix} \tag{5.3}$$

$$P_V = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & -1 \\ -2 & -2 & -2 & -2 & -2 \end{bmatrix} \tag{5.4}$$

After convolving the original image with each one of them, elements i and j of the output matrices G_H and G_V are summed up to have the magnitude of the image gradient $Grad$:

$$Grad(i,j) = \sqrt{G_H^2(i,j) + G_V^2(i,j)} \tag{5.5}$$

Eventually, the maximum value between the various element in



(a) Image to be analyzed (b) Horizontal gradient

(c) Vertical gradient (d) Total image gradient

Figure 5.3: Intermediate phases to image gradient computation

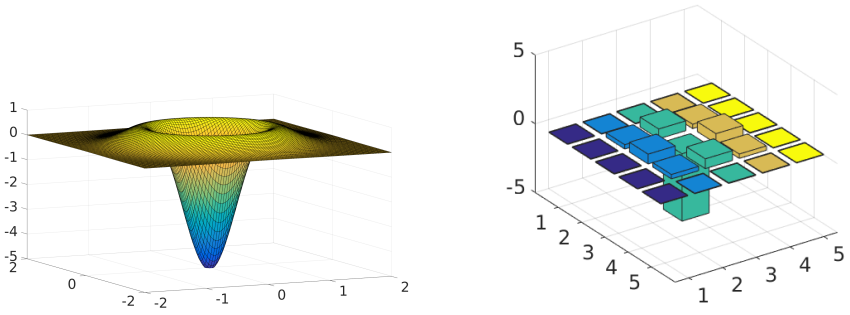
Eq. 5.5 is exploited to normalize the overall gradient matrix. It has been preferred to normalize with the current image *relative* maximum and not with the *absolute* maximum in order to have the most discrete possible values among the local gradient results. That is meaningful because the "reference" absolute value is already present in form of mean μ for the neural network.
Laplacian of Gaussian (LoG) filter [82] is a second order differential edge detector that combines a Gaussian smoothing filter with a Laplacian operator [77]. The Gaussian filter allows a noise reduction, while the Laplacian operator tracks the zero-crossing of the intensity function of the image, that represents maxima of such function: edges.

Analytically, it can be expressed as proposed in [76, 77]:

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{5.6}$$

where x, y are image coordinates, σ is the standard deviation of the Gaussian filter, defining the radius of the filter and therefore its sensitivity to reject noise. For the current hazard detection, being the image enough noise-free, a small $\sigma = 0.5$ have been selected. Such a function, often referred to as the "mexican hat" due to its shape, is shown in Fig. 5.4. Discrete approximated kernel to operate the LoG directly as a convolutional filter is thus:

$$\begin{bmatrix} 0.044792 & 0.046806 & 0.056407 & 0.046806 & 0.044792 \\ 0.046806 & 0.31675 & 0.71463 & 0.31675 & 0.046806 \\ 0.056407 & 0.71463 & -4.9048 & 0.71463 & 0.056407 \\ 0.046806 & 0.31675 & 0.71463 & 0.31675 & 0.046806 \\ 0.044792 & 0.046806 & 0.056407 & 0.046806 & 0.044792 \end{bmatrix} \tag{5.7}$$

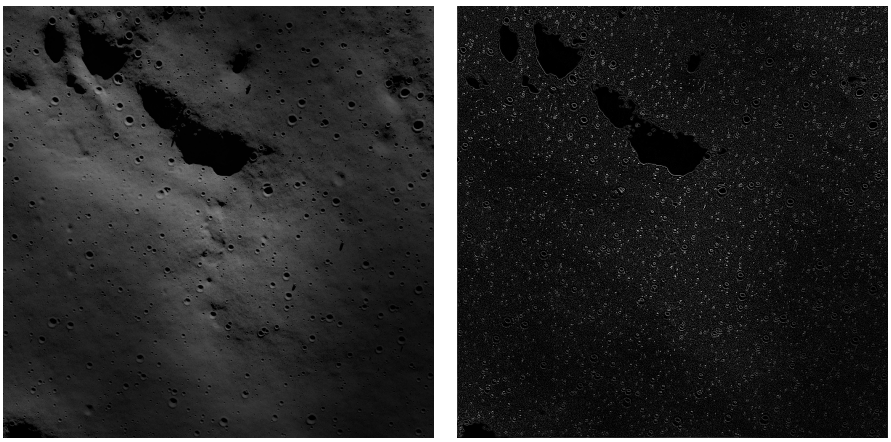


(a) Continuous function

(b) 5×5 kernel

Figure 5.4: Laplacian of Gaussian with $\sigma = 0.5$, representation as a continuous function (a) compared to the approximated discrete kernel used in the hazard detection system indices extraction.

Effects of the Laplacian of Guassian filter used for the hazard detection indices extraction on a lunar surface picture are depicted in Fig. 5.5.



(a) Image to be analyzed (b) LoG

Figure 5.5: Laplacian of Gaussian with $\sigma = 0.5$

As it is possible to spot in Fig. 5.5, Laplacian of Gaussian filtered image recognizes well edges, thus depicting craters and scraps borders with high accuracy. It results especially useful to identify very small features, such as small rocks and roughness.

In addiction to the indices extracted and processed from the camera image, also *Sun inclination angle* is passed to the feedforward artificial neural network, normalized between 0 and 1 dividing it by $\pi/2$. That value helps the ANN to understand the various features with different contrast levels: an image with a low Sun inclination angle will be characterized by net shadows and accordingly a greater overall sharpness. Hence, in the case of a general training process involving an input space where features at low Sun inclinations and features at high Sun inclination are balanced, the network will be facilitated in the recognition and hazard index assignment of areas characterized by great contrast, thus lower Sun inclination angle. So, providing that angle, the network can distinguish the same features at different light conditions.
Summarizing, *the overall input to the neural network is composed by 13 indices*:

- μ and σ for the three different mobile windows, thus 6 in total;
- LoG and gradient at three different image scales, thus 6 in total;
- Sun inclination angle.

It is important to underline that every input value is normalized to unity value in order to not incur in neurons saturation during the signal propagation among network's layers.

5.3 Neural Network simulation: hazard map computation

After the assembly of the various indices, the total input is fed into the *feedforward artificial neural network*. It consists in a simple 15 neurons single hidden layer, fully connected network. Output layer is a scalar value representing the hazard index of a the hazard map image. Activation functions exploited in the network are: *hyperbolic tangent* for the hidden layer, limiting the neurons output to [-1,1]; *logistic sigmoid* for the output layer, constraining the total output to the continuous set [0,1], where 0 represents a completely safe hazard map pixel, while 1 the most hazardous index. Thus for each input sample of 13 values, a single double number depicting the hazard value is computed by the network (Fig. 5.6). To relate each one of

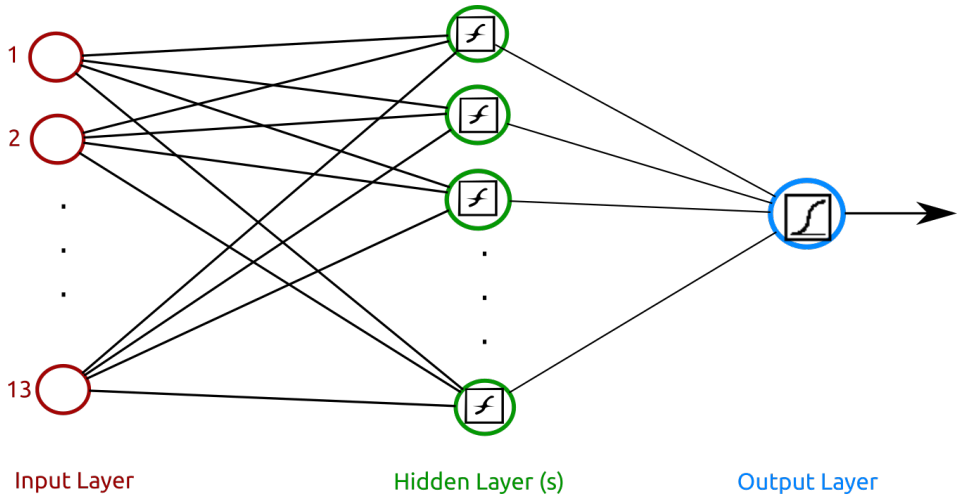


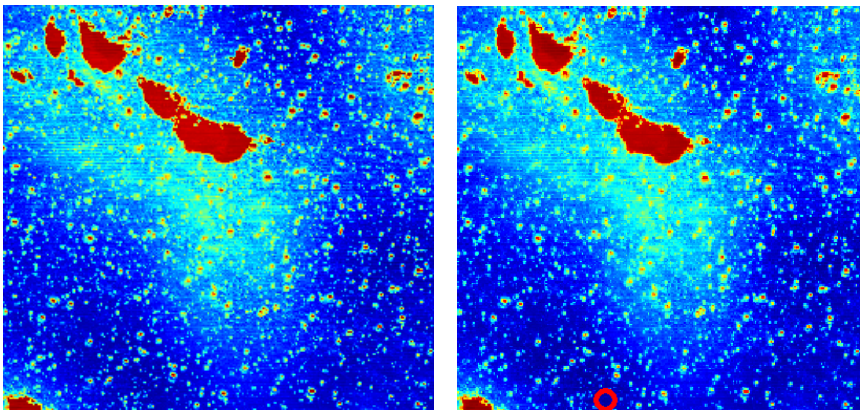
Figure 5.6: Feedforward ANN scheme. Activation functions are sketched for both the hidden and the output layer.

the output pixels with their neighborhoods, a very light blur filter is

implemented. Its convolutional kernel is:

$$\begin{bmatrix} 0.0075758 & 0.0075758 & 0.0075758 & 0.0075758 & 0.0075758 \\ 0.0075758 & 0.015152 & 0.015152 & 0.015152 & 0.0075758 \\ 0.0075758 & 0.015152 & 0.75758 & 0.015152 & 0.0075758 \\ 0.0075758 & 0.015152 & 0.015152 & 0.015152 & 0.0075758 \\ 0.0075758 & 0.0075758 & 0.0075758 & 0.0075758 & 0.0075758 \end{bmatrix} \quad (5.8)$$

It has been chosen such a light filter to prevent a too radical change of the pixels hazard index. In fact, an average or median filter performed on an unsafe pixel yields a relaxation of the pixel's hazard index, thus theoretically increasing the probability of false positive landing sites. As in can be noted in Fig. 5.7, the hazard map after the blurring filter is almost identical to the original.



(a) Hazard map, no blur (b) hazard map, blurred

Figure 5.7: Effects of the light blurring filter adopted on the hazard map are almost imperceptible to the eye.

The size of the neural network, which was much bigger in the original hazard detection system [67], has been drastically reduced without compromising performances: thanks to the enhanced indices extracted from the image, combined with a more significant training set, the neural network is facilitated in the various planetary surface features recognition. Note that a smaller network can reduce significantly the training time required.
Training of the ANN has been performed through C++ FANN library, exploiting the train and validation datasets described in Section 4.2. Training algorithm used is the *irprop* technique, the one proposed

by Christian Igel and Michael Hüsken as a faster variant of the resilient backpropagation (see Sec. 2.1.2), setting learning rate to 0.75 and momentum to 0.8 to achieve a quite aggressive learning (see the initial oscillations of the RMSE in Fig. 5.8). A custom early stop technique has been coded: validation dataset is used as usual to verify the RMSE of the network output with respect to the target, but not only the classical fixed number of consecutive iterations determines the reach for a local minimum of the validation error. Indeed, to prevent a too early stop due to the aggressiveness of the algorithm, also a validation considering the minimum of the error found in the overall training session is performed. This last validation is used to check if an absolute minimum of the error function is found during training.

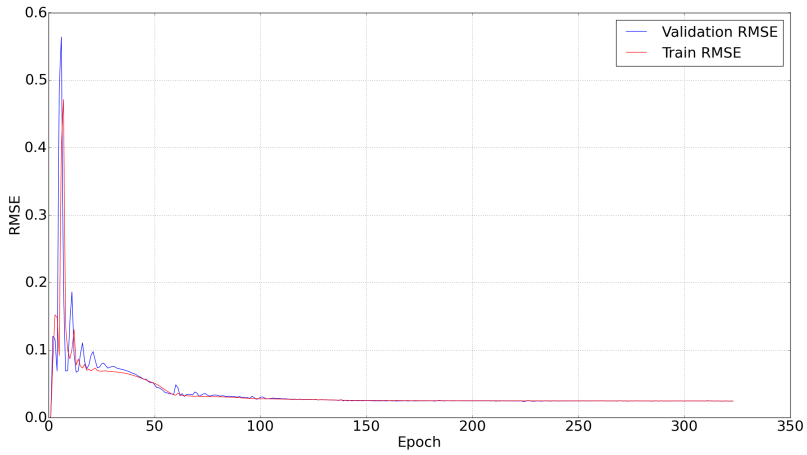


Figure 5.8: RMSE trend during training. It is possible to notice the initial oscillation of the RMSE due to the aggressive irprop algorithm settings adopted.

5.4 Landing site computation algorithm

Once the hazard map is available, the hazard detection system calls the *landing site selection* routine. The suitable sites for the lander are ranked according three parameters:

1. Minimum hazard index;

2. Maximum landing area;

3. Minimum distance from the Nominal Landing Site (NLS), in order to maximize the probability to find another landing site in case of necessary re-targeting.

The general scheme of the algorithm proceeds assigning three indices per landing site candidate, one per each of the just said parameters. Afterward, the three indices are merged to form the overall candidate's skill, and all the landing sites are sorted through the relative total hazard index in a global ranking.
To operate, the algorithm refers to the reference frame in Figure 5.9.

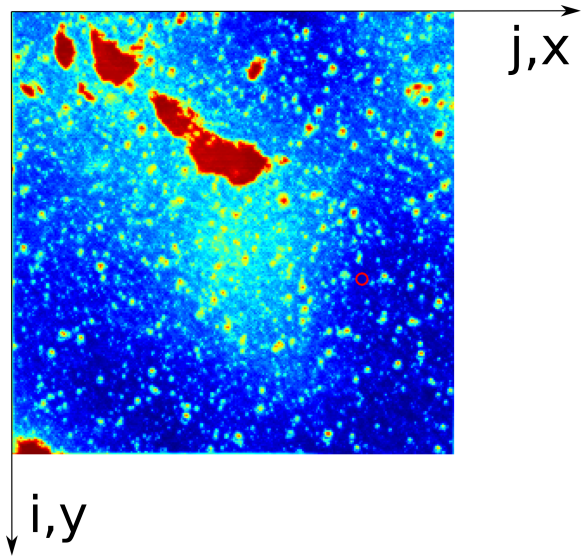


Figure 5.9: Image reference frame.

It is centered in the upper left corner of the image, aligned with image borders. The simple transformation in Equation 5.4 links the image space (pixels) with the actual surface space (meters):

$$[x\ y]^T = d_{res} s_s [i\ j]^T \tag{5.9}$$

where (x, y) represent the coordinates of a point expressed in meters, (i, j) the same point in pixels; d_{res} and s_s respectively the original image resolution in meters per pixel and the size in pixels of a side of the small mobile window used to extract the image indices. The knowledge of a sufficiently enough resolution is required to estimate landing sites properly, since their size must be compared with the actual dimension of the footprint of the lander, and thus the two values

must be in real world units mandatory. As stated in Section 5.1 also camera FoV and attitude are considered as available to the hazard detection system.
The algorithm develops through the following passages:

- The maximum safety threshold is applied to the hazard map, yielding a boolean map where True represents safe pixels, and False otherwise. This operation allows to easily and efficiently mark as unsafe all the False cataloged pixels. In the current hazard detection suite such a threshold is set to $\theta_{max} = 0.3$ for the ground truth hazard map: theoretically, it can be considered a $\theta = 0.33$ out of 1 value for a safe landing site, but due to the blurring filter present to relate nearby pixels (see Sec. 5.3), the unsafe areas are lightly lowered in hazard index due to the convolution operation with nearby safer pixels. *True* pixels, that are points such as $\theta_{ij} \leq \theta_{max}$, are classified as Candidate Landing Sites (CLS);

- For each True pixel, the *size score* $r_{CLS_{ij}}$ is calculated as the distance from the nearest unsafe (False) pixel. The higher the distance, the better the score;

- Considering the size of the lander and the navigation errors, a landing site must be chosen accordingly. Modeling the landing spot as a circle, its radius dimension is constrained as:

$$r_{CLS_{ij}} \geq r_{min} = \frac{d_{footprint}}{2} + e_{gnc} \tag{5.10}$$

  where r_{min} represents the minimum radius for the landing site, $d_{footprint}$ indicates the footprint diameter of the lander and e_{gnc} accounts for the imprecision of the Guidance and Navigation System at the desired confidence level. Every CLS that does not respect this constraint is excluded from the landing site candidates;

- The *Diversion score* $d_{CLS_{ij}}$ is computed for the remaining CLSs. It represents the distance from the NLS:

$$d_{CLS_{ij}} = \sqrt{(x_{CLS_{ij}} - x_{NLS})^2 + (y_{CLS_{ij}} - y_{NLS})^2} \tag{5.11}$$

  in which x_{NLS} and y_{NLS} represents the metric coordinates of the NLS, such as $x_{CLS_{ij}}$, $y_{CLS_{ij}}$ for the CLS in image frame position i, j.

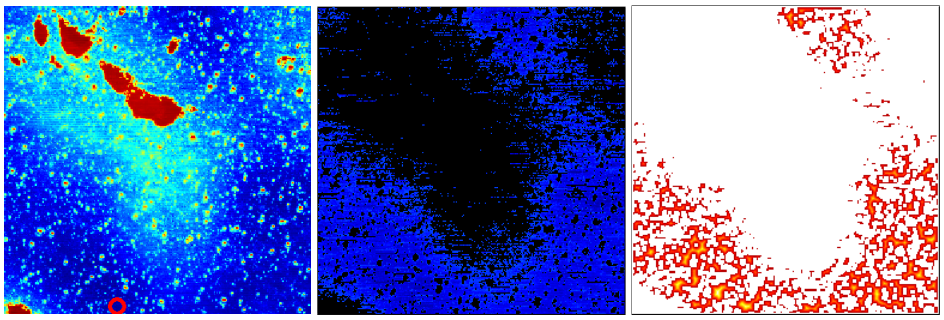
- The *Safety score* $z_{CLS_{ij}}$ is calculated as the average between all the hazard indices contained inside the circle centered in CLS$_{ij}$ with radius $r_{CLS_{ij}}$.

- Size score r_{CLS}, diversion score d_{CLS} and safety score z_{CLS} are normalized to 1 through the respective maximum value found in the current hazard map, and therefore they acquire the same magnitude one each other.

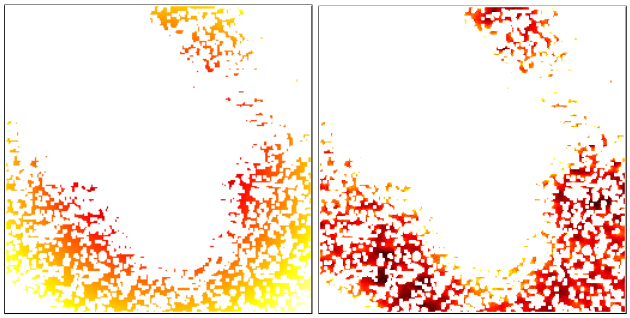
- The *Global score* $l_{CLS_{ij}}$ can now be computed:

$$l_{CLS_{ij}} = \mathbf{w}^T \begin{bmatrix} \tilde{r}_{CLS_{ij}} \\ 1 - \tilde{d}_{CLS_{ij}} \\ 1 - \tilde{z}_{CLS_{ij}} \end{bmatrix} \tag{5.12}$$

where **w** represents the vector of the weights that provides the flexibility to choose which of the three scores should be more considered during the landing site choice. It is clearly composed by three scalar values, each per score. The tilde ($\sim$) graphical sign represents a normalized variable.

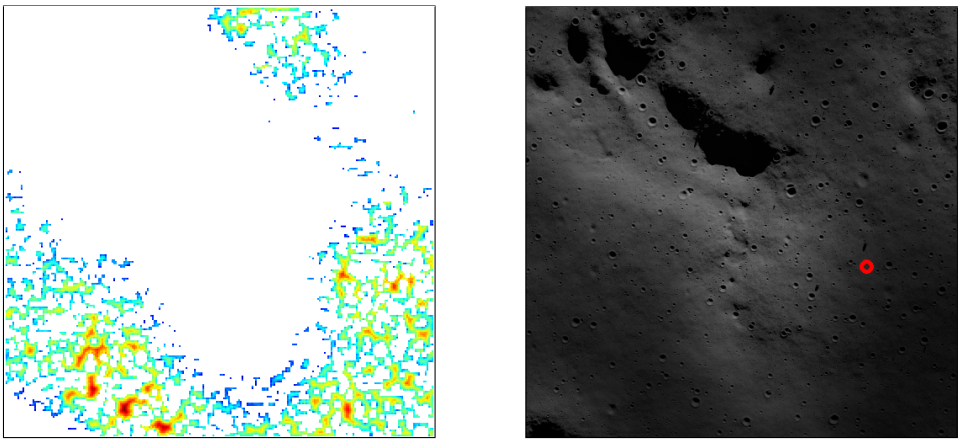


(a) Hazard Map (b) Threshold application (c) Size score



(d) Diversion score (e) Safety score

Figure 5.10: Intermediate phases of the landing site search algorithm: (a) Original hazard map. (b) Threshold of the original map with $\theta_{max} = 0.3$. *(c)* Size score *computation, thresholded with* $r_{CLS_{ij}} \leq r_{min}$. $d_{min} = 3\ m$, $e_{gnc} = 15\ m$. *(d)* Diversion score *computation: NLS is in set in the image center. (e)* Safety score *computation: average hazard index inside CLS radius*



(a) Candidates ranking (b) Landing site

Figure 5.11: Last phases of the landing site search algorithm: (a) Global score *computation: highest (red) area represents the best landing candidate (b) Landing site in the original lunar image.*

Once that the ranking of the various landing site candidates is available, the first element in such a vector represents the selected Target Landing Site (TLS), characterized by the highest score.

5.5 Landing Site Validation algorithm

In order to be able to evaluate scientifically and objectively performances of the hazard detection system, a landing site validation program has been developed. Such an algorithm is also necessary to compare the various hazard detector versions that have been developed in the past months. It is based on the landing site computation routine described in Sec. 5.4 and its main scope is the quantification of the following parameters:

- *True Positive* landing sites. Candidates generated through the neural network hazard map whose positions are safe also in the ground truth: they are correctly identified landing sites;

- *False Positive* landing sites. Candidates that instead are in an unsafe area of the ground truth hazard map, and therefore they represent a wrong identification performed by the neural network;

- *False Negative* landing sites. Areas considered unsafe by the neural network that are instead safe on the ground truth hazard map: they represent a wrong identification performed by the neural network.

- *First False Positive* occurrence among the landing site candidates ranking.

General criteria to validate the hazard detection system with are based on these general rules:

- if a landing site exists, it must be found;

- no false positive landing site should be computed, due to the fact that they represent a complete mission failure;

- false negatives are tolerated, but only if at least a correct identified landing site exists in order to not picture a critical treat to the mission.

To satisfy these requirements, it is therefore necessary that *the target landing site computed by the HD system is a true positive.* Moreover, in order to assure the possibility for the spacecraft to re-target towards a new landing site in case of necessity, it is mandatory that at least the first 2 candidates in the landing site ranking are true positives for any test set image. The optimization of the performances of the hazard detection architecture, also taking into account re-targeting needs, is discussed in Sec. 5.5.1.
The algorithm to validate the landing sites found by the neural network is structured as the following:

- Ground truth hazard map of the analyzed lunar surface image is computed with the corresponding landing sites[1]. Hazard threshold is set to 0.3/1;

- Hazard detection system's hazard map is calculated through the neural network, with the corresponding landing sites[2]. Hazard threshold can be tailored to optimize the results;

[1] From now on: "*True landing sites*"
[2] From now on: "*ANN landing sites*"

- A *True map* (TMAP) is generated, thresholding the ground truth hazard map with the threshold 0.3, that is:

$$\text{TMAP} = \text{HMAP}_{GT} \leq \theta_{max} \tag{5.13}$$

  where HMAP_{GT} is the ground truth, θ_{max} is the maximum hazard index allowed. TMAP represents the safe pixels onto which the hazard detection software must find the landing sites in order to provide a suitable landing spot for the lander;

- *True Positives and False Positives computation.* Every ANN landing site candidate is "tested" onto the TMAP: if it is completely included, that means that every pixel of its radius is where the thresholded ground truth is *True*, it is classified as True Positive (TP). Otherwise, even if a single pixel is not part of the TMAP, it is considered as a False Positive (FP);

- *False Negatives computation*. The ANN computed hazard map is thresholded with the θ_{ANN} selected to create the FNMAP:

$$\text{FNMAP} = \text{HMAP}_{ANN} \leq \theta_{ANN} \tag{5.14}$$

  where HMAP_{ANN} corresponds to the neural network generated hazard map. In FNMAP, True value is assigned where the local hazard index is less or equal than the custom threshold. Then, each true landing site candidate is "tested" onto the FNMAP: if it is located entirely in a False pixel set of the FNMAP, it is classified as a False Negative (FN);

- *First False Positive*. ANN landing site candidates are sorted trough their hazard indices. From the first in ranking –that represents the TLS–, a cycle analyzes every candidate after the other until the first False Positive is identified.

5.5.1 Optimization of the landing site candidates ranking

Performances of the HD system depend on many internal parameters. Besides choosing different architectures of the image indices extraction routine, increasing the size of the ANN, adding SOM or other different items, it is possible to achieve the maximum performances of a fixed architecture optimizing the hazard index threshold adopted for the computed map and the three landing site search program weights

introduced in Section 5.4.
In general, performances of HD system will be the better the higher the number of true positives identified, the worse the higher number of false positives occurred. Such a performance is adjusted acting on the hazard threshold allowing a greater or minor number of hazard map pixels to be included in the landing site candidates. On the other hand, optimization of the weights allows to modify the found landing site candidates sorting in the ranking. Therefore, assuming at least one TP and one FP have been found, an action on the weights will directly influence the position of the first false positive found in the ranking of the landing site candidates. Ideally, the best possible performances coincide with the totality of landing sites computed to be true positives, no false positives, no false negatives. Being in practice almost impossible to achieve such a perfection, the optimization process should maximize the number of true positives, minimizing the amount of false positives and false negatives. Additionally, in the case of FP occurrence, *it should be at least in position 3 in the overall candidates ranking*, in order for the spacecraft to be able to re-target towards a safe backup landing site in case the selected is not possible to be reached for. A detailed analysis about the performances optimizations is done in Section 6.2.

5.6 Discarded Architectures

The most representative attempts made during the survey of the best image indices extraction routine are presented to the reader. In Section 6.4 their performance comparison will be discussed. Between the various tries, both architectures with and without Kohonen networks have been tested. Nevertheless, the feedforward network used in all the architectures counts a single hidden layer with 15 neurons, in order to have a coherent comparison among all of them. Moreover, after the network processing of the image extracted inputs, the same very slight blur filter (see Eq. 5.8) is applied to the hazard map generated, in order to relate nearby pixels explicitly, and not only through the internal weights of the neurons.

Version A

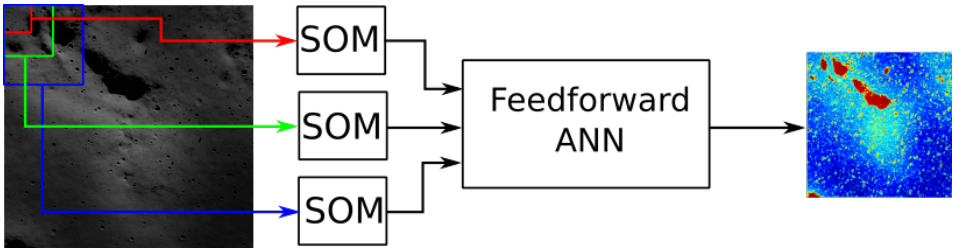


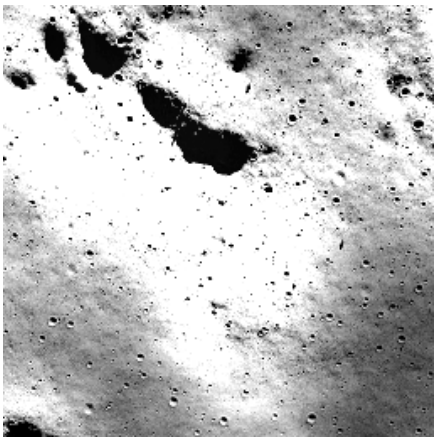
Figure 5.12: Version A, architecture.

This version is inspired by ORACLE+ architecture: the three Kohonen networks with size 10×10 are maintained prior the feedforward multilayer ANN, but the secondary ANN is removed (Fig. 5.12). The indices used in this version are exactly the same exploited in the original architecture of the hazard system [67], but half of mobile windows dimensions are used to obtain an hazard map image resolution of 256×256 px instead of 128×128:

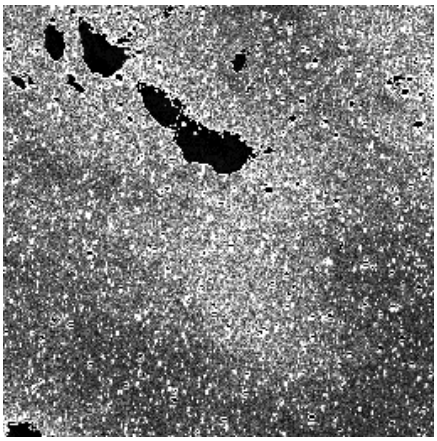
- For window size **4**×**4** px:
  - directional *mean*
  - directional *standard deviation*
- For window size **8**×**8** px:
  - directional *mean*
  - directional *standard deviation*
- For window size **16**×**16** px:
  - directional *mean*
  - directional *standard deviation*

To prevent neurons saturation during the signal propagation in the neural network, mean is normalized to [0,5], while standard deviation to [-10,10]. As an example, in Figure 5.13 the two statistical indices computed for the central column of the smaller mobile window are shown.

(a) Original Image



(b) Mean



(c) Standard deviation

Figure 5.13: Directional mean and standard deviation computed on the central column direction for 4×4 mobile window.

To remind the reader the meaning of *directional* computed index, the sketch depicting the various directions is proposed again in Fig. 5.14.

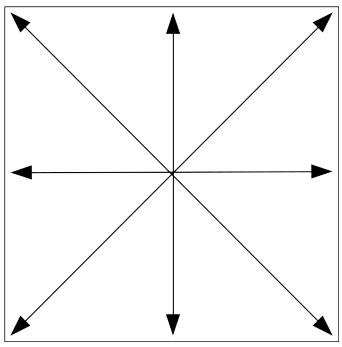


Figure 5.14: Directions used to compute mean and standard deviation in a single mobile window in the original hazard detection system.

Version B

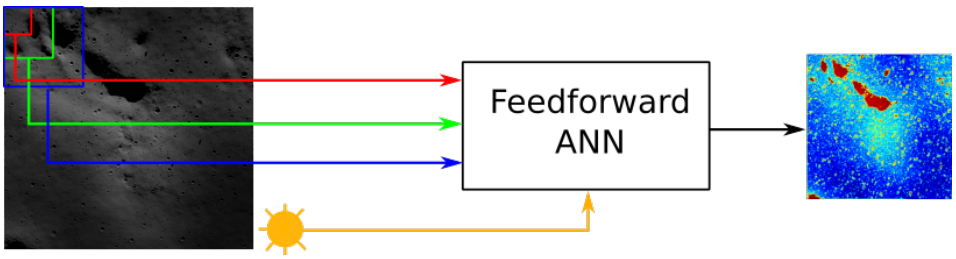


Figure 5.15: Version B, architecture.

No Kohonen nets are exploited in this version, as the indices extracted from the image are directly fed into the ANN. Such indices are:

- For window size **4**×**4** px:
  - *Mean* of the whole mobile window pixels
  - *Standard deviation* of the whole mobile window pixels
- For window size **8**×**8** px:
  - *Mean* of the whole mobile window pixels
  - *Standard deviation* of the whole mobile window pixels
  - *Image gradient* through Sobel filter
- For window size **16**×**16** px:
  - *Mean* of the whole mobile window pixels

  - *Standard deviation* of the whole mobile window pixels
  - *Image gradient* through Sobel filter

- Sun inclination angle

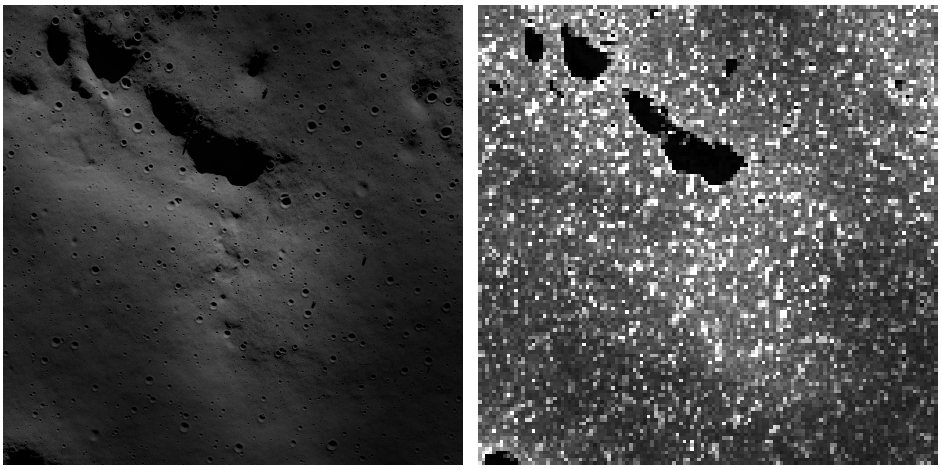
The horizontal and vertical gradients (grad_x and grad_y) through the Sobel filter are achieved performing convolution on the image with kernels:

$$H = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad V = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{5.15}$$

and the total gradient (grad) show in Fig. 5.16 for every image pixel in position i, j is straightforward:

$$\text{grad(ij)} = \sqrt{\text{grad}_x(i,j)^2 + \text{grad}_y(i,j)^2} \tag{5.16}$$

To prevent neurons saturation, mean is normalized to [0,1] standard



(a) Original image (b) Image gradient

Figure 5.16: Image gradient computed through Sobel filter

deviation to [-1,1], gradient to [0,1], Sun inclination angle to [0,1]. Note that the gradient theoretically has domain up to infinity. Therefore to normalize the Sobel output it has been exploited the output generated by a white line in black field, that express the maximum gradient possible for a 8 bit gray scale image. It resulted equal to 1020. In any case, thanks to the robustness of the neural networks,

even if such normalization is not extremely precise the difference in ANN output is negligible. What really matters to obtain a reliable and efficient network is to be coherent in training and ANN execution, that is using the same scale. For what concern Sun inclination, it is normalized through the maximum possible Sun elevation of 90°. Every window processed with the Sobel filter is downsampled to 1×1 px through a series of Gaussian pyramids.

Version C

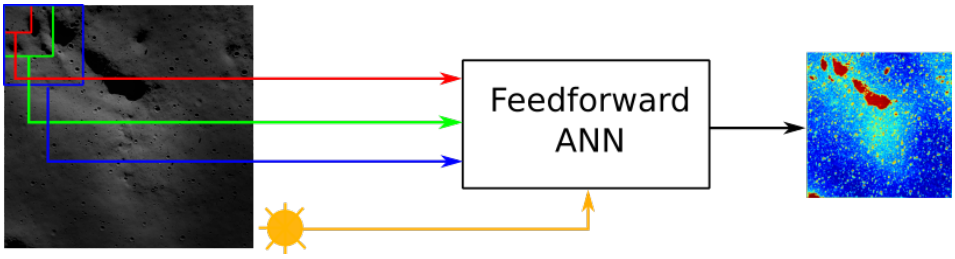


Figure 5.17: Version C, architecture.

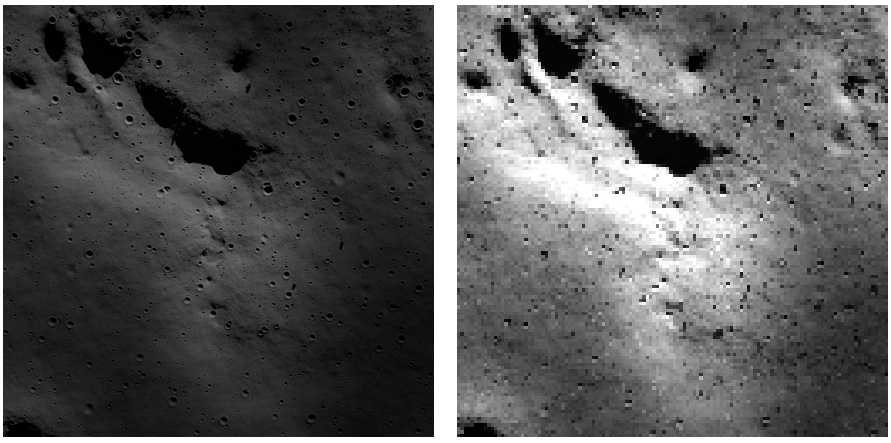
No Kohonen nets are exploited in this version, as the indices extracted from the image are directly fed into the ANN. Such indices are:

- For window size **4**×**4** px:
  - Directional *Mean* of the mobile window pixels as in Version A
  - Directional *Standard deviation* of the mobile window pixels as in Version A
- For window size **8**×**8** px:
  - Directional *Mean* of the mobile window pixels as in Version A
  - Directional *Standard deviation* of the mobile window pixels as in Version A
  - *Image gradient* through Sobel filter
  - *Laplacian of Gaussian* with $\sigma = 0.5$
- For window size **16**×**16** px:

  - Directional *Mean* of the whole mobile window pixels
  - Directional *Standard deviation* of the whole mobile window pixels
  - *Image gradient* through Sobel filter
  - *Laplacian of Gaussian* with $\sigma = 0.5$

- Sun inclination angle

Normalization is performed in this version too: mean to [0,5], standard deviation to [-10,10], gradient to [0,10], Laplacian of Gaussian to [-10,10], Sun inclination angle to [0,1]. Image gradient normalization has been handled as in version B. Following the same approach used for the gradient for the LoG, the normalization coefficient computed is: 1250.71. The effect of the LoG filter is shown in Fig. 5.18[3].



(a) Original Image (b) LoG

Figure 5.18: Effects of Laplacian of Gaussian filter on the original image.

[3]In order to have a visualization of the LoG effects its values have been converted to positive because of the -png format limitations. Due to that, such picture does not represents the actual input fed into the ANN, but only a representation suitable to the human eye

Version D

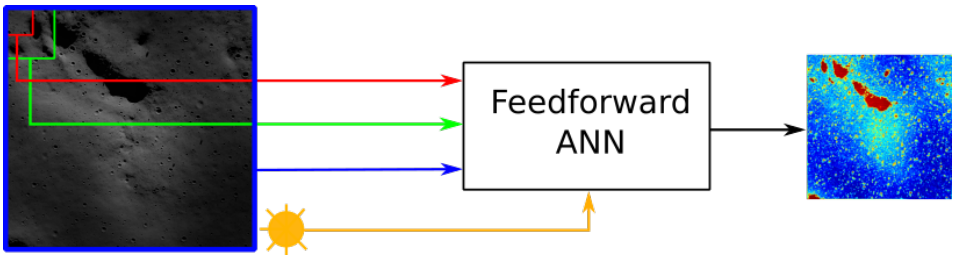
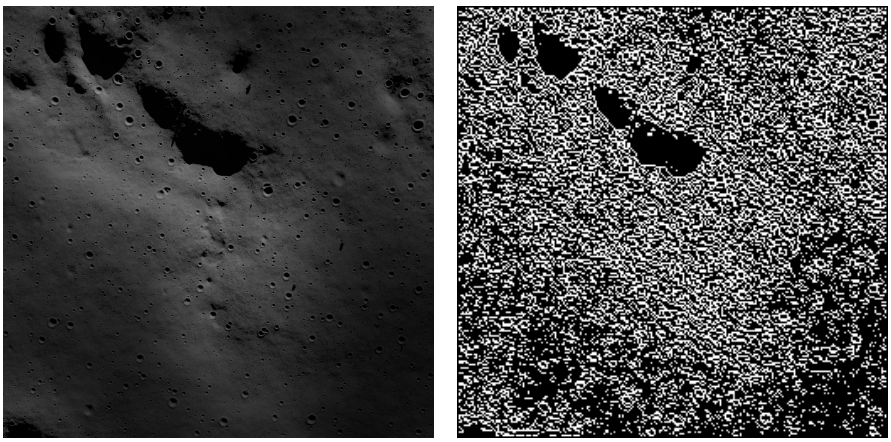


Figure 5.19: Version D, architecture.

This is the first version to introduce *global* indices over the traditional *window-based* indices. No SOM has been used. Indices extraction procedure is the following:

- Window-based:
  - *Mean* of the whole mobile window pixels for windows sizes of 4×4, 8×8, 16×16
- Global:
  - Binary map generated through edge detector based on LoG with $\sigma = 0.8$
  - *Standard deviation* with 5×5 kernel
- Sun inclination angle

Global indices, as base image to process, use the map generated through the mean computed with the window-based approach. Binary image for the edges is calculated skeletonizing the output of the gradient after applying the threshold σ. This way, value 1 identifies an edge, 0 otherwise (Fig. 5.20).



(a) Original Image (b) Edges with LoG

Figure 5.20: Effects of the edge detector based on LoG filter with $\sigma = 0.8$.

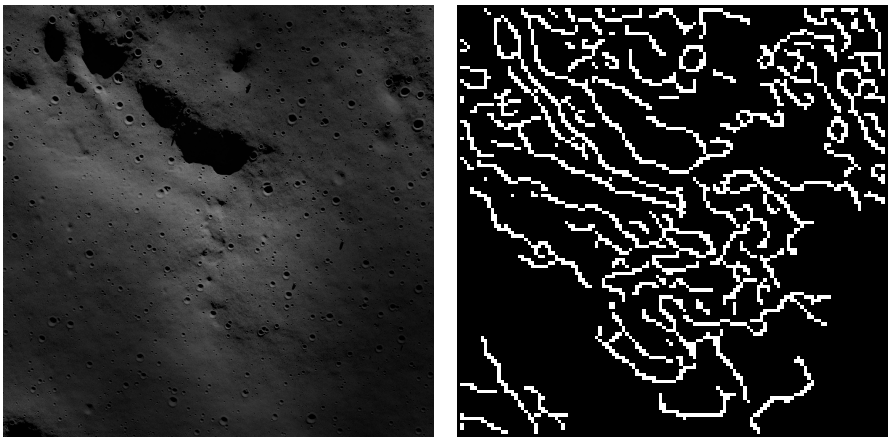
Image topological skeleton is exploited in order to have thin edges on the output image, even if the zero crossing borders of the objects in the picture are wide. Standard deviation in this version of the HD system is computed across the 5×5 neighborhood of the considered pixel and it is normalized between [-1,1]. Mean domain is transformed to [0,5].

Version E

An enhanced version D, where the architecture remains the same as the one in Figure 5.19. Once again no Kohonen nets, and edge detector uses a Canny filter[83] instead of a Laplacian of Gaussian, while the standard deviation is computed with a smaller 3×3 kernel. Hence, the indices:

- Window-based:
  - *Mean* of the whole mobile window pixels for windows sizes of 4×4, 8×8, 16×16
- Global:
  - Binary map generated through edge detector based on Canny filter with $\sigma_{low} = 0.04$ and $\sigma_{high} = 0.1$
  - *Standard deviation* with 3×3 kernel
- Sun inclination angle

Canny filter, also known as the optimal edge detector in literature[83, 84], works smoothing the input image with a Gaussian filter, then applying a gradient -such as Prewitt or Sobel filters- and thresholding out spots of the image characterized by a non-maximum gradient magnitude (non-maxima suppression). Great novelty of the Canny based edge detector is the use of two threshold to obtain an hysteresis: after the first gradient reduction through the *low threshold* σ_{low}, if the magnitude of the pixels derivative is higher than the *high threshold* σ_{high}, it should be an edge. In the case of a pixel is in between the two thresholds, it is set to zero unless it is directly topologically connected on the image to a pixel with a gradient higher than σ_{high} [83]. Edge detection with the selected Canny filter is shown in Figure 5.21. Standard deviation filter is the same used in version D with



(a) Original Image (b) Edges with Canny

Figure 5.21: Effects of the edge detector based on Canny filter with $\sigma_{low} = 0.4$ *and* $\sigma_{high} = 0.1$

a smaller kernel equal to 3×3 px and it is normalized to [-10,10]. Mean domain is bounded to [0,5]. Clearly the edge detector output is binary as in version D.

Version F

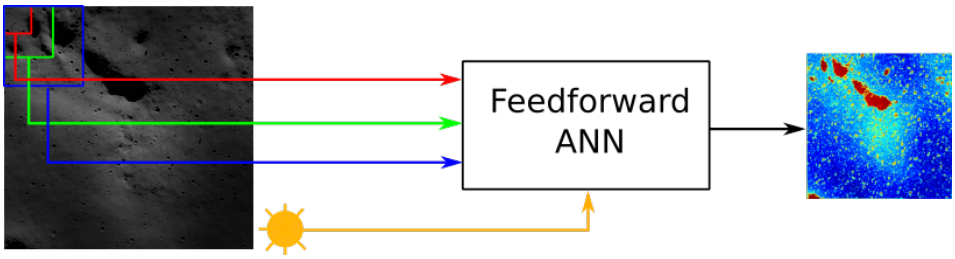


Figure 5.22: Version F, architecture.

This version proposes a lighter variant with respect to version C: in intermediate mobile window LoG and gradient are remove to investigate the effects of maintaining only such filters on the largest scale. Moreover, mean and standard deviation are eliminated from the large window. To clarify, indices extraction proceeds as:

- For window size **4**×**4** px:
  - Directional *Mean* of the mobile window pixels as in Version A
  - Directional *Standard deviation* of the mobile window pixels as in Version A
- For window size **8**×**8** px:
  - Directional *Mean* of the mobile window pixels as in Version A
  - Directional *Standard deviation* of the mobile window pixels as in Version A
- For window size **16**×**16** px:
  - *Image gradient* through Sobel filter
  - *Laplacian of Gaussian* with $\sigma = 0.5$
- Sun inclination angle

Mean and standard deviation are normalized respectively to [0,5] and [-10,10]. Gradient to [0,10] and LoG to [-10,10] through the same procedure used in version C.

Version G

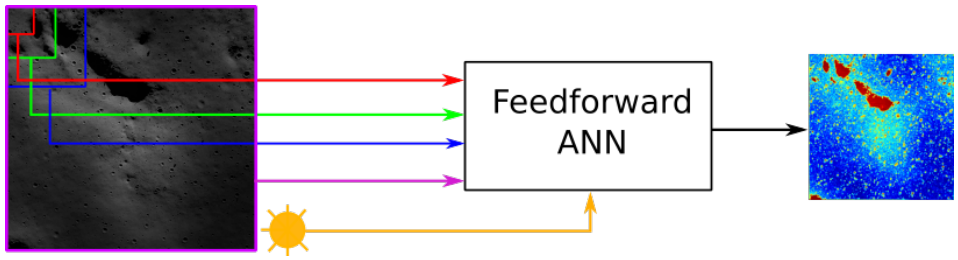


Figure 5.23: Version G, architecture.

Version that includes also the overall mean of the image:

- For window size **4**×**4** px:
  - *Laplacian of Gaussian*
- For window size **8**×**8** px:
  - *Mean* of the whole window pixels
  - *Gradient* through Sobel filter
  - *Laplacian of Gaussian*
- For window size **16**×**16** px:
  - *Gradient* through Sobel filter
- Mean computed on the whole
- Sun inclination angle

Sobel filter domain: [0,10], Laplacian of Gaussian: [-10,10], mean: [0,5]. It has been inserted the mean calculated on the whole image to investigate the feedforward neural network capability to handle different albedos. Unfortunately, the datasets available are composed by images with terrain optical characteristics quite similar among them, and therefore, no actual testing was possible to quantify the capability of the network to recognize hazard index for similar features with different albedos.

Version H

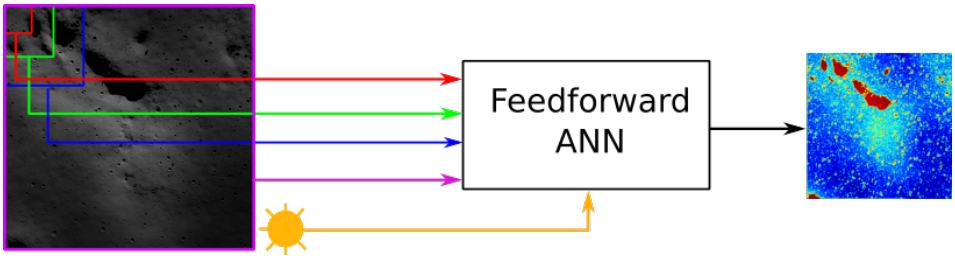


Figure 5.24: Version H, architecture.

Almost identical to Version G, but it has been removed the gradient in the intermediate window:

- For window size **4**×**4** px:
  - *Laplacian of Gaussian*
- For window size **8**×**8** px:
  - *Mean* of the whole window pixels
  - *Laplacian of Gaussian*
- For window size **16**×**16** px:
  - *Gradient* through Sobel filter
- Mean computed on the whole image
- Sun inclination angle

Sobel filter domain: [0,10], Laplacian of Gaussian: [-10,10], mean: [0,5].

Version I

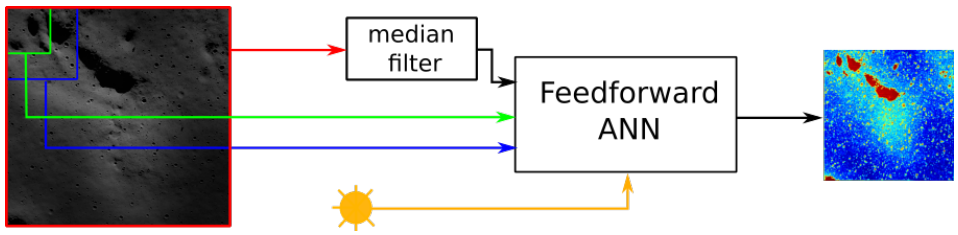


Figure 5.25: Version I, architecture.

What differences mainly this version is the use of a *median filter* during the indices extraction routine.

- Window based:
  - For window size **8**×**8** px:
    - *Mean* of the whole window pixels
    - *Standard deviation* of the whole window pixels
  - For window size **16**×**16** px:
    - *Mean* of the whole window pixels
    - *Standard deviation* of the whole window pixels
- Global, computed after application of a median filter with kernel size 5×5 and image downsample through Gaussian pyramids down to 256×256:
  - *Standard deviation* with a 3×3 kernel.
  - Binary map generated through *edge detector* based on LoG with $\sigma = 0.5$
- Global, computed after application of a median filter with kernel size 5×5 and image downsample through Gaussian pyramids down to 128×128, 64×64:
  - Binary map generated through *edge detector* based on LoG with $\sigma = 0.5$
- Sun inclination angle

The choice to insert a median filter has been done to smooth downsampled image in order to investigate the capability to identify vast sloping regions in the image, probably the most difficult surface property to recognize. Indeed, the median filter associated with the Gaussian pyramid tends to blur small scale surface features like craters and terrain roughness, leaving therefore only higher scale characteristics.

Version L

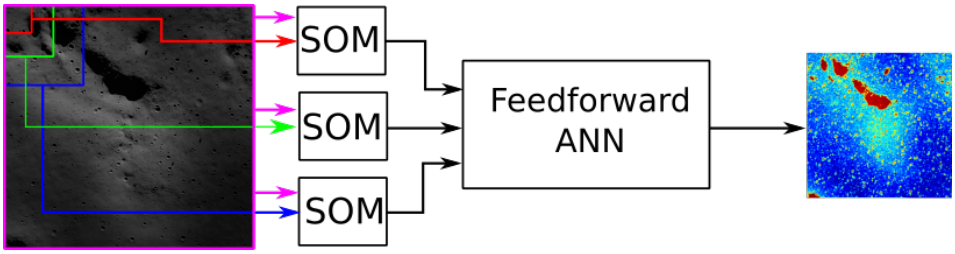


Figure 5.26: Version L, architecture.

Kohonen networks are proposed again in order to assess the performances of the current hazard detection system architecture's indices extraction routine (see Sec. 5.2), if it is assisted by SOM input space reduction. Two versions will be proposed, differing in which input is preprocessed by Kohonen maps. This first sees all the indices being fed to three SOMs, one per window size (or resulting image downsample), as shown in Figure 5.25. The indices are:

- Window based:
  - For window size **4**×**4** px:
    - *Mean* of the whole window pixels
    - *Standard deviation* of the whole window pixels
  - For window size **8**×**8** px:
    - *Mean* of the whole window pixels
    - *Standard deviation* of the whole window pixels
  - For window size **16**×**16** px:
    - *Mean* of the whole window pixels
    - *Standard deviation* of the whole window pixels

- Global
  - *Laplacian of Gaussian* with a 5×5 kernel and $\sigma = 0.5$.
  - *Image gradient* through Prewitt filter.
- Sun inclination angle

LoG and image gradient are both downsampled to 256×256, 128×128, 64×64 px to match dimensions of the output of the window based indices extraction. Mean is normalized to [0,5], standard deviation to [-10,10], image gradient to [0,1] and LoG to [-1,1]. Note that the global indices are normalized through the *relative* maximum value, intending the highest in that specific image and not in absolute. That allows the output of the filters to be always extended to the maximum also if the current picture to analyze has low gradients or LoG, situation very frequent with high Sun elevation images. The three Self-Organizing Maps have size of 10×10 neurons each.

Version M

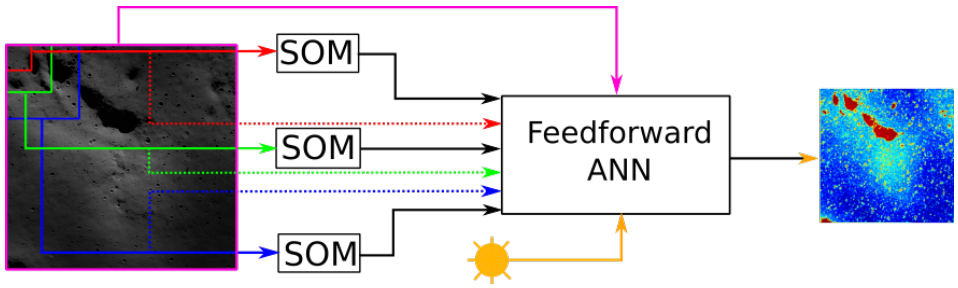


Figure 5.27: Version M, architecture.

Again a version exploiting Self-Organizing Maps. The difference with the previous one is only the fact that not all the inputs are preprocessed by Kohonen nets, as hereby specified:

- *Mean* and *standard deviation* are fed into SOM
- *LoG* and *image gradient* are directly processed by the feedforward ANN

All the three Kohonen networks are composed 10×10 neurons.

Chapter 6

Performances assessment: tests and results

PERFORMANCES *of the hazard detection system will be hereafter objectively quantified. A statistical analysis is accomplished on the test set in order to validate the system, assessing its capability to yield* safe *results in the various operative conditions a lander is subjected to. At this point, the hazard detection system presented in this thesis is almost complete: the ANN has been trained with a large dataset, its parameters have been optimized and a validation routine to check objectively the correctness of the found landing sites is ready. Moreover, a comparison with the previous architecture iterations shown in Chapter 5 is presented to justify the current selection. Firstly, the hazard maps generated through the ANN are related with the ground truths crafted following the procedure described in Chapter 4. Afterward, correctness of computed landing site candidates is discussed and quantified.*

6.1 Hazard maps

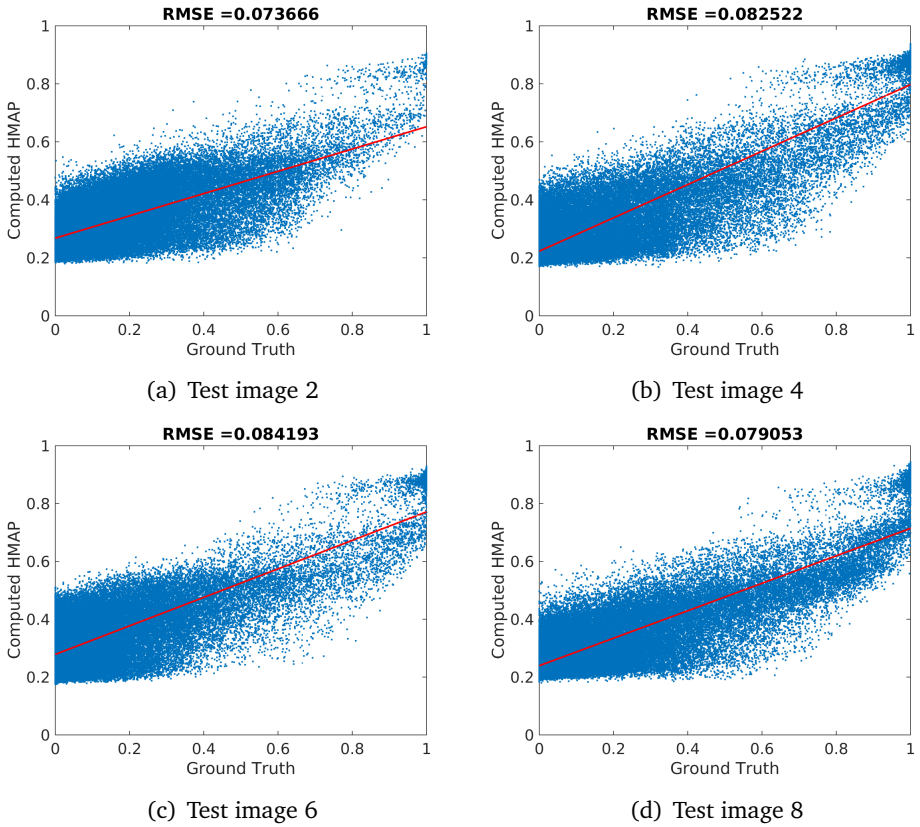
Evaluation of the accuracy of the neural network output is performed through the *Root Mean Square Error (RMSE)* statistical index, comparing the computed hazard map pixels with respect to the corresponding ground truth ones. Clearly, to coherently assess hazard map goodness, such as any other parameter of merit concerning the neural network, it is necessary to refer to the test dataset. As specified in Chapter 4, the test set is completely uncorrelated with respect to

the pool of data exploited for the training. In this way the ability of the system to give a proper response also in conditions not explicitly considered during the project phase is assured, and the avoidance of over-fitting phenomena is verified. As parameter of merit for the evaluation of the quality of an hazard map, the *Root Mean Square Error (RMSE)* has been adopted. It is defined as:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{N}} \tag{6.1}$$

in which N is the number of pixels of the hazard map, $\hat{y}_i$ represents the i-th pixel's intensity, y_i is instead the i-th point of the regression line computed between the points of the ground truth and the computed hazard map. The less the RMSE is, the better result the ANN has achieved, because it means that the computed hazard map is well approximating the ground truth with similar corresponding pixels intensities. In this case, the points in the plot ground truth versus computed hazard map tends to stick nearby the linear regression line. Indeed, if such a low RMSE is achieved, even if the regression line does not represents the bisecting line of the plot, it is just a matter of neural network's hazard map threshold tailoring, that, speaking in terms of the plot, represents the interception with the ordinate axis and the slope of the regression line.
In Figures 6.1 and 6.2, regression lines per each test image are proposed to the reader. The 8 test dataset images are listed following the Appendix B sorting. Quantitatively speaking, the RMSE results are shown in Table 6.1.

A low RMSE is in general observed, with a total average among the test dataset of 7.1%. In particular, and it will be seen also in the majority of the discarded architectures, the 80° Sun inclination angle images have a lower RMSE: an average of 6.2%, opposite to the low Sun elevation dataset that registers a 8.0%. In Figure 6.2 it is possible to notice that the network understands the hazard threshold upper limit, that does not goes up to 1 due to the lack of shadows. A general behavior that is observed is the positive y-axis interception point of the regression line: it means that for completely safe areas the network is anyway assuming that –even if it is lower than the threshold of 0.28– they do not deserve the safest index. Such a highly conservative behavior could be the reason of the great number of false



(a) Test image 2

(b) Test image 4

(c) Test image 6

(d) Test image 8

Figure 6.1: RMSE computation: images with 15° Sun elevation in the test dataset.

negative landing sites occurring during computation, as it will be seen in Section 6.2.

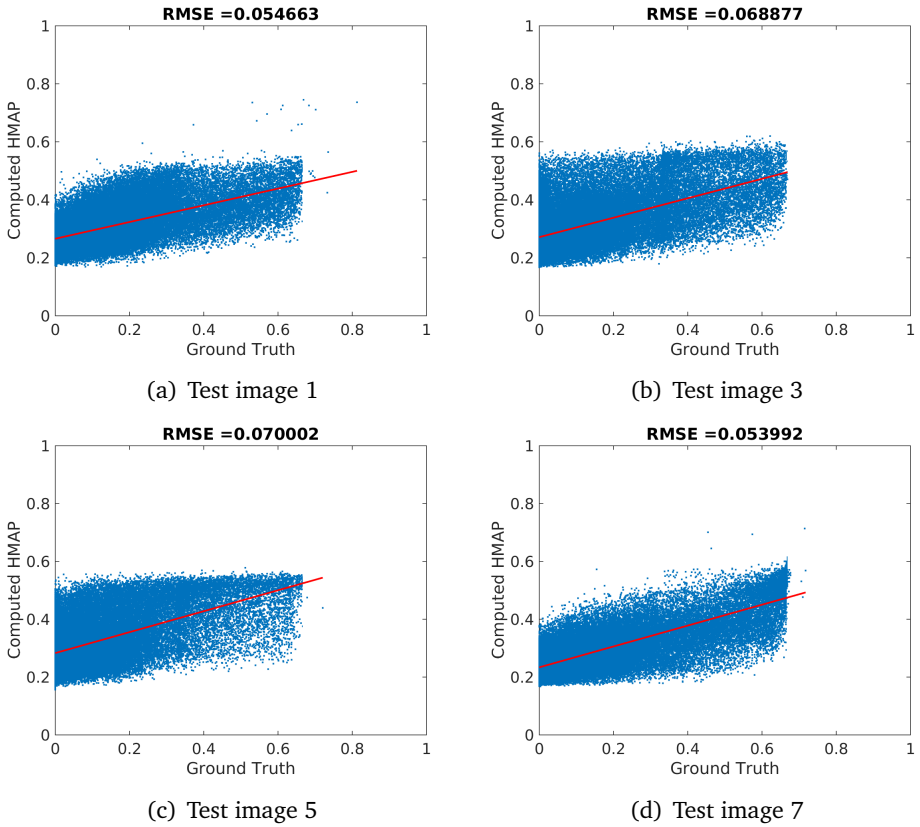
6.2 Landing site selection

The performances assessment proceeds with the quantifications of the correct and incorrect landing sites found. As described in Chapter 5 such evaluation depends on the number of True Positives, False positives, False Negatives occurred. Two indicators are introduced:

- *Correct Prediction Index (CPI)*. It is defined as:

$$\text{CPI} = \frac{\text{nTP}}{\text{nTP} + \text{nFP}} \tag{6.2}$$

  where nTP is the number of True Positives encountered, nFP the number of False Positives. Note that the denominator nTP+nFP



(a) Test image 1

(b) Test image 3

(c) Test image 5

(d) Test image 7

Figure 6.2: RMSE computation: images with 80° Sun elevation in the test dataset.

equates the total number of landing sites found by the neural network. CPI is bounded between [0,1] where 1 means that the hazard detection system landing site candidates are all correct, 0 the opposite – the higher the better –;

- *Missed Landing sites Value (MLV)*. Introduced to quantify how many actual landing sites the neural network does not recognizes as such, it is defined as:

$$\text{MLV} = \frac{\text{nFN}}{\text{nFN}+\text{nTP}} \tag{6.3}$$

where nFN is the number of False Negatives landing sites, thus the lower the MLV the better the performance. The MLV parameter is secondary in importance with respect to the CPI, because a high number of False Negatives does not represent a direct threat to the mission, whereas a high number of False Positives does.

Test image	**Sun elevation [$^\circ$]**	**RMSE [-]**
1	80	0.054663
2	15	0.073666
3	80	0.068877
4	15	0.082522
5	80	0.070002
6	15	0.084193
7	80	0.053992
8	15	0.079053
Mean (Sun 15°)		**0.079859**
Mean (Sun 80°)		**0.061884**
Mean (all)		**0.070871**
Max (all)		**0.084193**
Min (all)		**0.053992**

Table 6.1: Hazard map RMSE.

To perform the numerical computations, a footprint $d_{footprint} = 3$ m and a navigation error $e_{gnc} = 15$ m at $3\ \sigma$ confidence have been considered.
The presence of multiple parameters to be optimized concurrently requires theoretically a *Multi-Objective Optimization (MOO)*. In particular, an elitist variant of the NSGA-II genetic algorithm [85] has been approached as first attempt. Due to the fact that computational cost of the optimization grows linearly with the number of objectives [86], a smart choice allows to drastically reduce the onerous optimization time. With this considerations in mind, the first optimization program have been executed with the following objectives:

- *maximization* of the minimum first positive position in the candidates ranking between the test dataset images, normalized with the total number of found landing site candidates in the corresponding image;
- *maximization* of the minimum Correct Predictions Index (CPI) value between the test dataset images.

As it is possible to notice, both of the two objectives are bounded to domain [0,1]. A population of 200 individuals has been generated to perform the MOO.

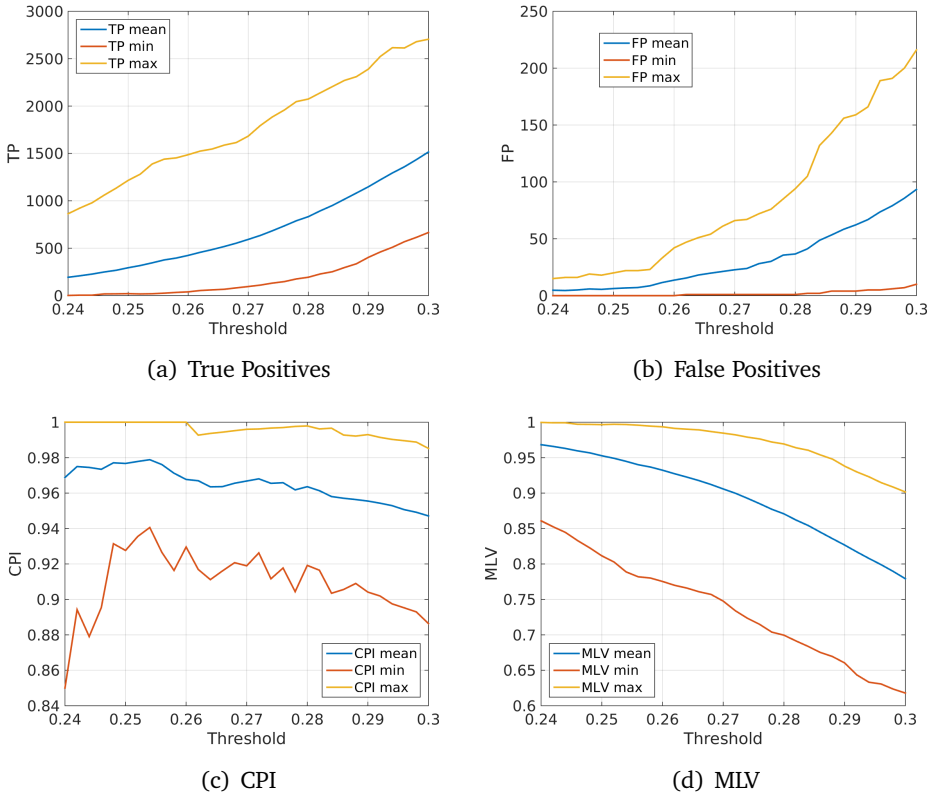
For what concerns the constraints of the optimization it has been selected:

- hazard index threshold of the computed hazard map between 0.25 and 0.30;
- sum of the landing site weights equal to 1;
- landing site selection weights between 0 and 1.

The optimization process, performed in Matlab® environment on the personal computer described in Section 1.4, have been run more than once. In any case, more than one day was requested to reach for a solution, which was not in any case satisfactory, due to the fact many local stationary points are present in the CPI function, as it is shown in Figure 6.4. Because of this CPI function irregularity, individuals of the genetic algorithm population do not converge properly, being them "fooled" by such local minima (maxima). An increasing of the individuals quantity would have probably improved the convergence quality, but no enough powerful computer was available to run such a onerous multi-objective optimization.
To overcome this issue, a two step optimization has been performed:

- a sensitivity analysis has been done on various performance parameters, such as CPI, MLV and ratio between TP and FP, versus the threshold of the computed hazard map. This allowed to assess very quickly how the performances of the hazard detection system change with the hazard threshold and to choose consequently the best value;
- a single-objective optimization (SOO), where variables to optimize are only the landing site weights, and the objective is the position of the first false positive found, since it is the only parameter that landing site weights modify. Genetic algorithm has been ditched in favor of Particle Swarm Optimization (PSO) algorithm because of the faster convergence rate experimented on the SOO.

For what concerns the first, the various important trends resulting from the computations on the usual test dataset images are depicted in Figure 6.3. As expected, a clear increasing in the average of true



(a) True Positives

(b) False Positives

(c) CPI

(d) MLV

Figure 6.3: Trend of various HD system performances changing the threshold of the computed hazard map.

positives amount is observed: raising the threshold, a greater quantity of landing sites candidates is found. Nevertheless, it is possible to state the same for the false positives: percentage increasing of the mean is even higher than the true positive, as shown by the higher derivative of the FP curve in Figure 6.3b. In absolute values the growth of true positives minimum between test images outnumbers the corresponding false positives increasing up to threshold of 0.28 (Figure 6.4), where the curve slope becomes negative. Indeed, whereas the TP increasing ratio is quite constant (Fig. 6.3a), the FP curve is characterized by a non negligible derivative increasing at 0.28 of threshold, clearly visible in Figure 6.3b. Therefore, *it has been decided to opt for a computed hazard map threshold of 0.28*.

One could also argue why it has not decided to set such a threshold to 0.25, value at which the false positives are less. The reason is clear looking at MLV and TP trends (Fig. 6.3), in particular the MLV max-

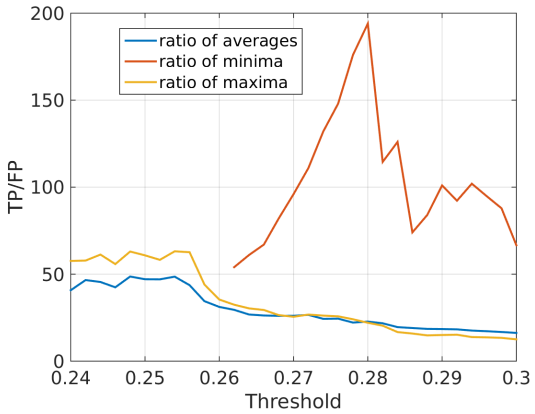


Figure 6.4: Trend of TP/FP considering average, maxima, minima of TP and FP values among the test dataset images.

ima and the TP minima curves: to allow the hazard detection system to be able to spot a significant number of landing sites in any condition, the maximum MLV of about 0.99 present at threshold 0.25 have been considered too low. The same can be understood looking at the true positive minima trend. Moreover, the CPI observes not significant variations from 0.25 to about 0.28, meaning that the percentage of true positive found with respect to the total –that is increased with 0.28– is almost the same.
Results for the analysis run on the test set are summarized in Table 6.2. An average CPI of over 0.96 shows that the HD system's landing site candidates are correct for the vast majority. In particular, the lowest CPI recorded amounts to only 0.919 for test image 8, showing that in any case at least almost the 92% of the landing sites found are correct. Surprisingly, the hazard detection system scored an higher CPI on the 80° Sun inclination test subset, which is actually characterized by lower light gradients and therefore less sharp features representations, resulting harder to understand in theory. The MLV is instead in general a quite high 0.87, with the 15° Sun inclination test subset showing a lower 0.83, that is 0.08 points better than the performances on the 80°. Hence more landing sites are in general understood with high contrast shadowy images, as one could expect. With the less sharp intensity variation of high Sun elevation images, the network tends to spread the high hazard indices of dangerous surface features in the neighbor pixels, resulting conservative in terms of

Test image	**Sun elevation [°]**	**CPI [-]**	**MLV [-]**
1	80	0.989796	0.969275
2	15	0.953125	0.907793
3	80	0.997890	0.919898
4	15	0.956642	0.699551
5	80	0.992126	0.906528
6	15	0.979257	0.840152
7	80	0.920916	0.854177
8	15	0.919192	0.868450
Mean (Sun 15°)		**0.952054**	**0.828986**
Mean (Sun 80°)		**0.975182**	**0.912470**
Mean (all)		**0.963618**	**0.870728**
Max (all)		**0.997980**	**0.969275**
Min (all)		**0.919192**	**0.699551**

Table 6.2: Hazard detection system performances.

safety, but it considers many safe small spots nearby higher hazard index places as unsafe. That is the reason of an higher MLV for the 80° subset. In Figures 6.8 to 6.15 at the end of the chapter are shown the hazard maps computed by the HD system on the test set.
In any case, the most important performance to evaluate is the position of the first false positive encountered in the landing sites candidates ranking. As specified in Section 5.5.1, in order to guarantee the possibility to re-target the lander towards a new landing site if the first target was discovered, the first false positive should be at least 3rd in the ranking. Many PSO optimizations with various initial conditions have been performed in order to find different combinations of weights that yield an optimal result. The triplets of calculated optimal landing site search weights are listed in table 6.3. Selecting one of the rows of table 6.3 guarantees that *the most dangerous position of the first false positive in the landing site candidates ranking among the images of the test dataset is 3*.

Although for the HD system performances on the test dataset the selection of one of the rows in Table 6.3 is equivalent in terms of first positive occurrence in the landing sites ranking, it has been decided to opt for the triplet [0.63636 0.27273 0.090909], as it represents a good balance between the first two coefficients, relative to the land-

Landing site radius	Mean hazard index	distance from NLS
0.8	0.1	0.1
0.81818	0.090909	0.090909
0.72727	0.18182	0.090909
0.75	0.16667	0.083333
0.76923	0.15385	0.076923
0.63636	0.27273	0.090909
0.66667	0.25	0.083333
0.69231	0.23077	0.076923
0.71429	0.21429	0.071429
0.61538	0.30769	0.076923
0.64286	0.28571	0.071429
0.66667	0.26667	0.066667
0.57143	0.35714	0.071429
0.6	0.33333	0.066667
0.625	0.3125	0.0625

Table 6.3: List of optimal landing site search function weights: each row is a combination of optimal weights.

ing site radius and to the mean hazard index, and has the highest value among the third, that weights the distance from the Nominal Landing Site (NLS). Indeed the others have equal or lower weight for the distance to the NLS that would yield a negligible contribution of the third weight on the candidates ranking. With such weights, the first false positive occurrence in the landing site candidates ranking for the test dataset images is listed in Table 6.4

Test image	**Sun elevation [$^\circ$]**	**Position of first FP**
1	80	177
2	15	41
3	80	298
4	15	4
5	80	149
6	15	58
7	80	3
8	15	23
Mean (Sun 15°)		31.5
Mean (Sun 80°)		156.75
Mean (all)		94.125
Max (all)		298
Min (all)		3

Table 6.4: First false positives in the landing site candidates ranking for the test dataset.

Due to the fact that the optimization has been performed on the test set, the same on which performances have been computed, another DEM not present in the test dataset has been used to confirm the quality of the system. Three images of such a DEM have been generated at 10°, 45°, 80 ° to obtain a complete quantification at different light conditions. Their respective ground truth and computed hazard maps are shown in Figures 6.5 to 6.7. As it is possible to

Image	**Sun elevation [°]**	**RMSE [-]**
1	10	0.082593
2	45	0.057524
3	80	0.060675
Mean (all)		**0.066931**
Max (all)		**0.082593**
Min (all)		**0.057524**

Table 6.5: Hazard map RMSE for the DEM outside the test dataset exploited to validate the performances after the landing site search weights optimization.

notice in Table 6.5, less than 6.7% in average was recorded for the RMSE of the three images, with a peak of 8.2% for the one characterized by the lowest Sun elevation angle. As expected, these values are perfectly in line with the test dataset results in Table 6.1.

For what concerns the performances about the landing site candi-

Test image	**Sun elevation [°]**	**CPI [-]**	**MLV [-]**
1	10	0.942308	0.858472
2	45	0.987399	0.846702
3	80	0.978610	0.917406
Mean (all)		**0.96944**	**0.87419**
Max (all)		**0.987399**	**0.917406**
Min (all)		**0.942308**	**0.846702**

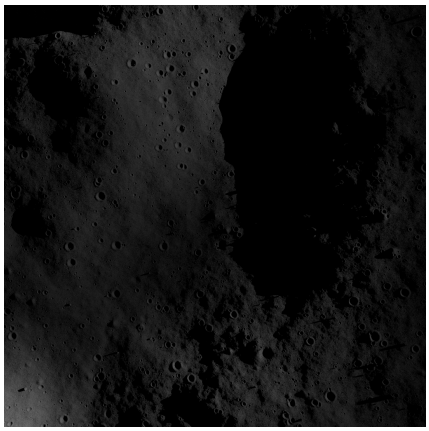
Table 6.6: Hazard detection system performances for the DEM outside the test dataset exploited to validate the performances after the landing site search weights optimization.

dates selection, Table 6.6 shows very satisfactory CPI values always over 0.94, with a peak of 0.99 for the 45 ° Sun elevation angle image. The average MLV of 0.87 is almost the same of the test dataset performances records in Table 6.2. Testing the HD system with the optimal

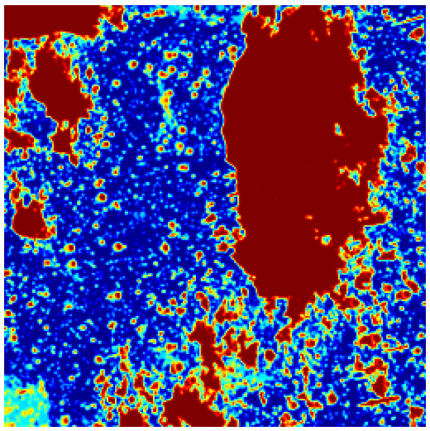
[0.63636, 0.27273, 0.090909] landing site selection weights, first false positives occurrences in the three images are equal or greater than 15th position, as summarized in Table 6.7. It is possible to consider this test a success.

Test image	**Sun elevation [°]**	**Position of first FP**
1	10	15
2	45	138
3	80	90
Mean (all)		81
Max (all)		138
Min (all)		15

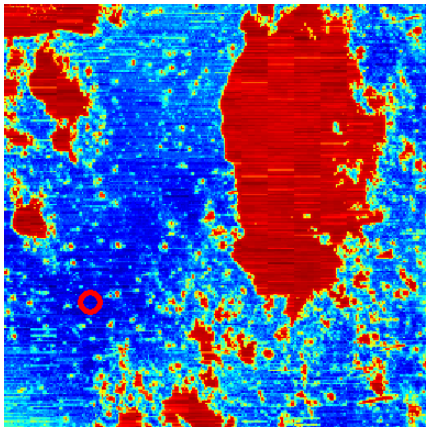
Table 6.7: First false positives in the landing site candidates ranking for the DEM outside the test dataset exploited to validate the performances after the landing site search weights optimization.



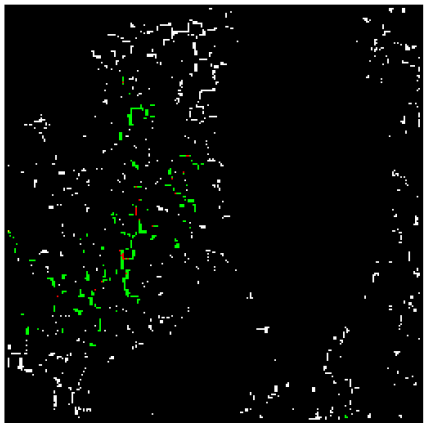
(a) Lunar surface



(b) Ground truth

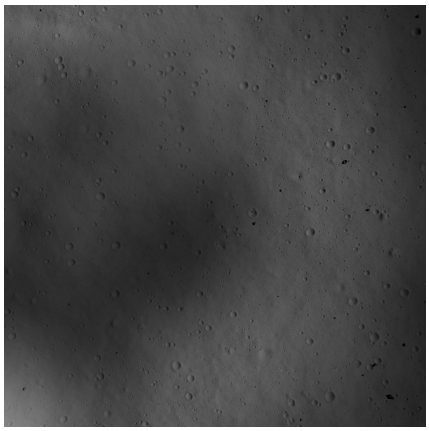


(c) Computed hazard map

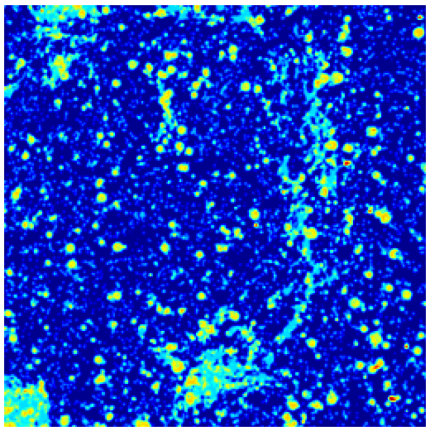


(d) Landing Sites Validation

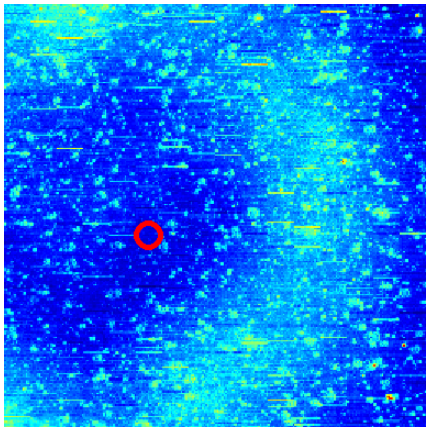
Figure 6.5: Ground truth and computed hazard map comparison for a DEM not present in the test dataset rendered at 10 ° Sun elevation angle. In (d), red means False Positive, green True Positive, white False Negative



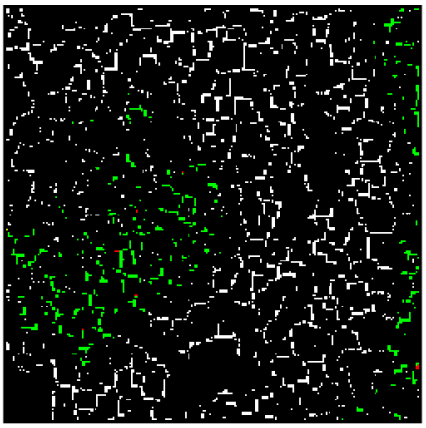
(a) Lunar surface



(b) Ground truth

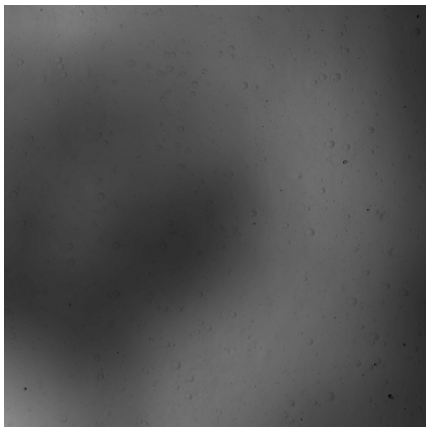


(c) Computed hazard map

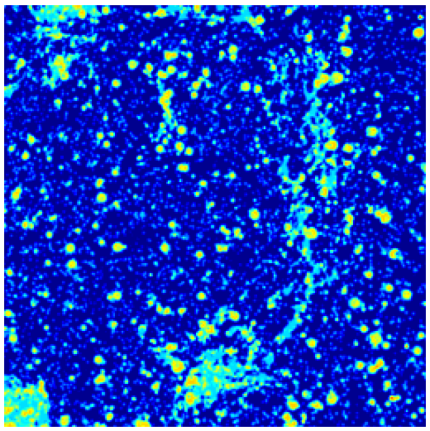


(d) Landing Sites Validation

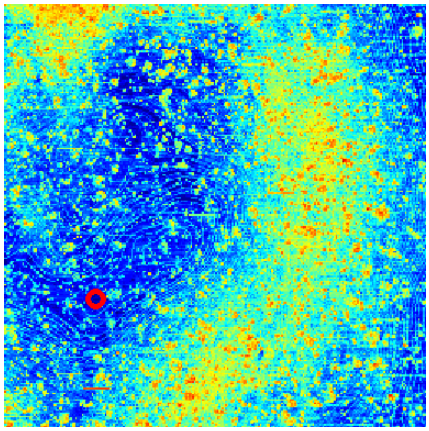
Figure 6.6: Ground truth and computed hazard map comparison for a DEM not present in the test dataset rendered at 45 ° Sun elevation angle. In (d), red means False Positive, green True Positive, white False Negative



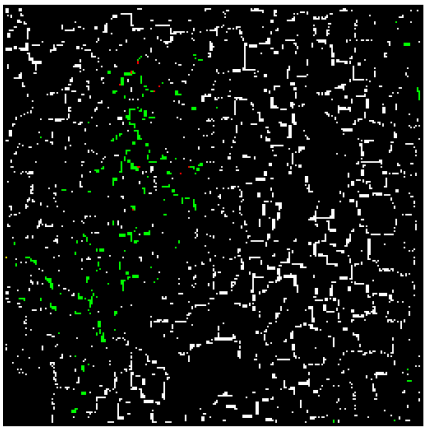
(a) Lunar surface



(b) Ground truth



(c) Computed hazard map

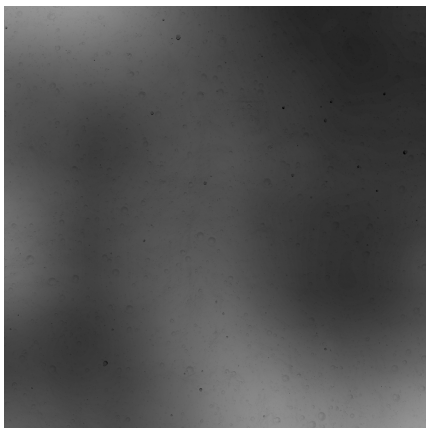


(d) Landing Sites Validation

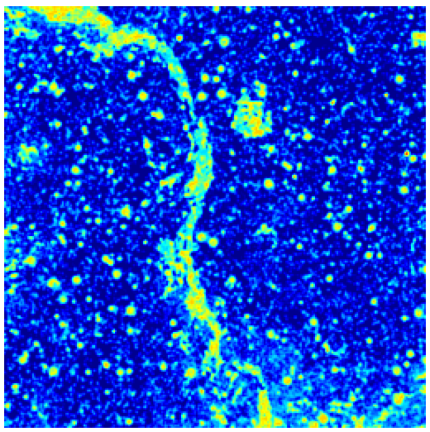
Figure 6.7: Ground truth and computed hazard map comparison for a DEM not present in the test dataset rendered at 80 ° Sun elevation angle. In (d), red means False Positive, green True Positive, white False Negative

6.3 Computational performances

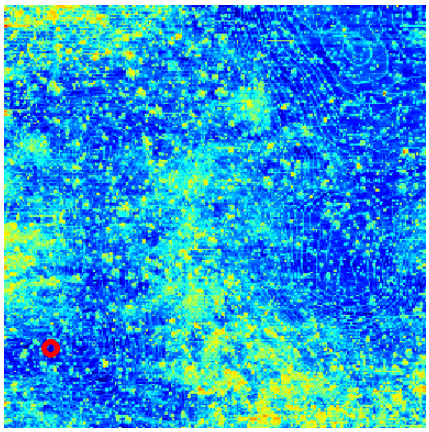
The importance of a computationally efficient hazard detection system is clear: a real-time analysis has to be executed during the descent of the lander. Hence, particular care has been invested in reducing the execution time of the HD software. On the other hand, no compromises with respect to the HD results have been made. As introduced in Section 1.4, it has been exploited the high efficiency of C++ programming language to develop a "flight-like" version of the hazard detection system. All the calculations have been performed on a Intel® Core™ i7-4712HQ CPU clocked @ 2.3 GHz paired to 16 Gb of DDR3 memory. The machine runs 64 bit Ubuntu 15.04 GNU/Linux as operative system. To properly estimate the computational weight of the proposed HDA system, a profiling analysis has been carried out. *Gperftools* [87], a tool released by Google under BSD license, has been selected as main profiler; as verification, obtained result have been crosschecked with the Linux GNU's *time* command [88]. In order to avoid modern processors' automatic multi-thread splitting of a running process, the system has been forced to run in single-thread configuration by properly setting processor affinity. In each profiling test, the hazard detector runs in a cycle for 1000 times, while the sampling frequency of the Google profiler has been set to 250 Hz (the highest possible value) to maximize the precision in runtime estimation. Gperftools registered 56 666 hits at 250 Hz, for a total time of 226.66 s, while for the *time* command the correspondent CPU time resulted 235 s. Taking into account the possible overhead that can affect measurements differently with the two methods, the values are comparable and the hazard detection system mean runtime can be considered of 230 ms. The main bottleneck has been identified in the indices extraction routine, that requires about the 65% of the total. Such measurements were expected, since image process algorithms are computationally demanding, especially when performed in a serial fashion. In actual highly parallel hardware units, such as Field Programmable Gate Arrays (FPGA), are extremely faster in those sort of calculations.



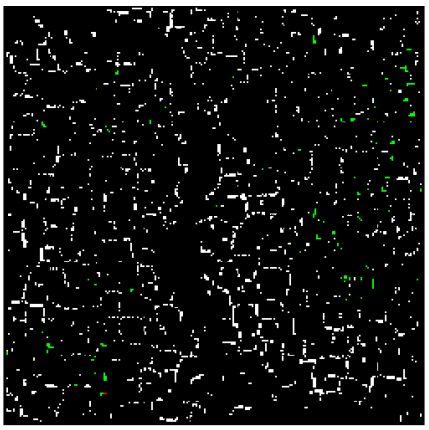
(a) Lunar surface



(b) Ground truth

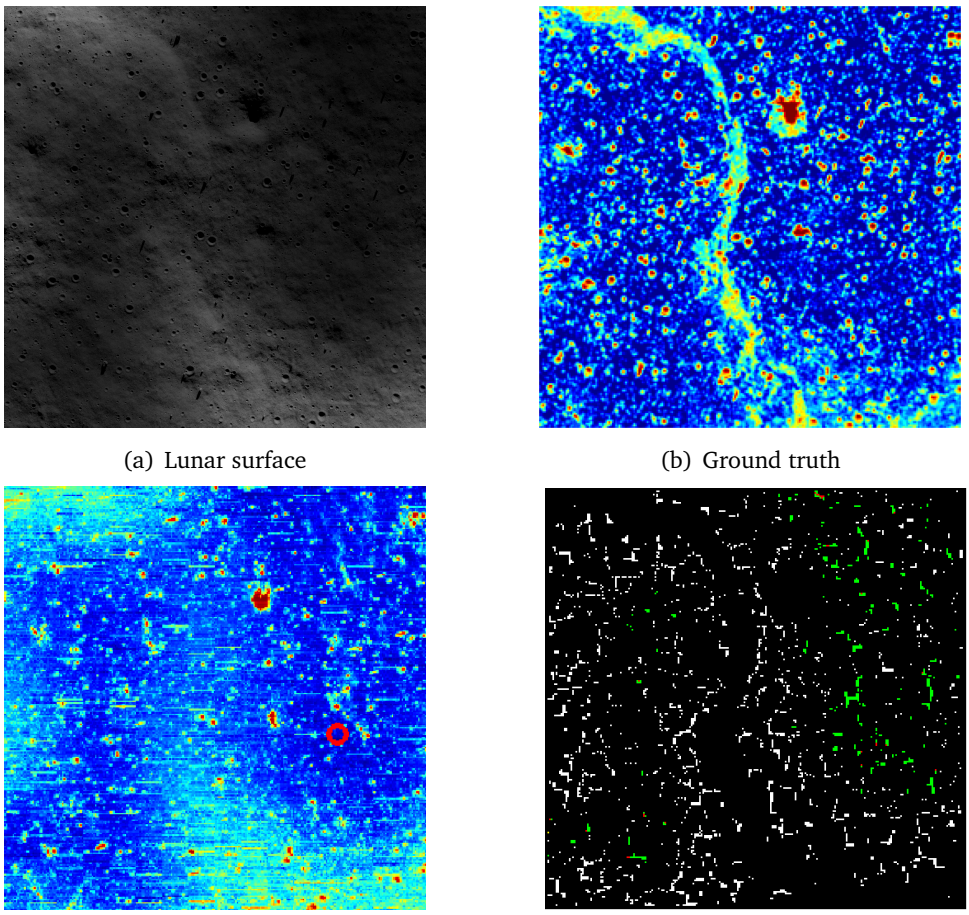


(c) Computed hazard map



(d) Landing Sites Validation

Figure 6.8: Ground truth and computed hazard map comparison, test image 1. In (d), red means False Positive, green True Positive, white False Negative



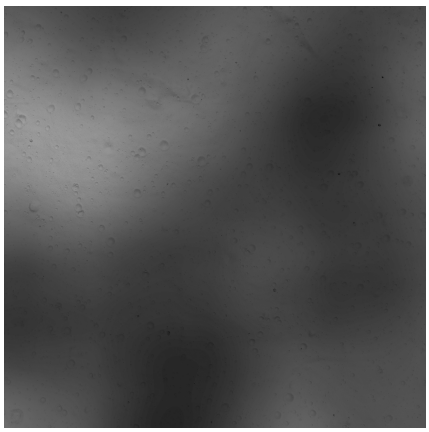
(a) Lunar surface

(b) Ground truth

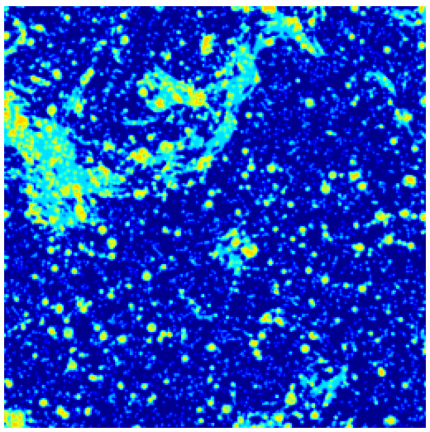
(c) Computed hazard map

(d) Landing Sites Validation

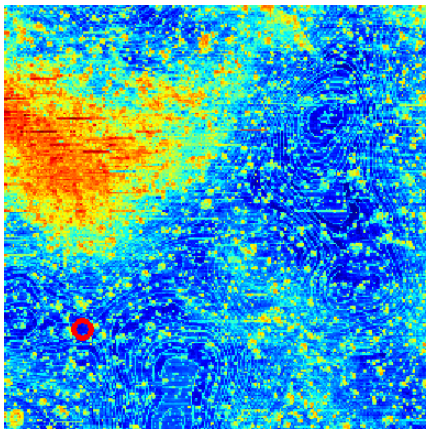
Figure 6.9: Ground truth and computed hazard map comparison, test image 2. In (d), red means False Positive, green True Positive, white False Negative



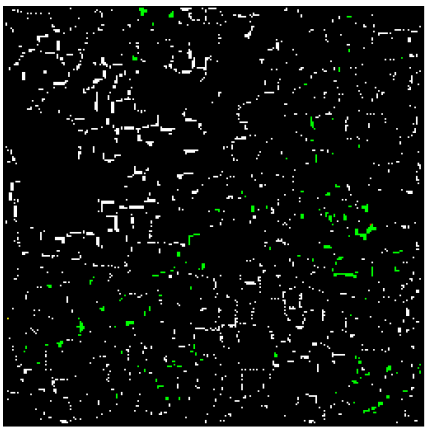
(a) Lunar surface



(b) Ground truth

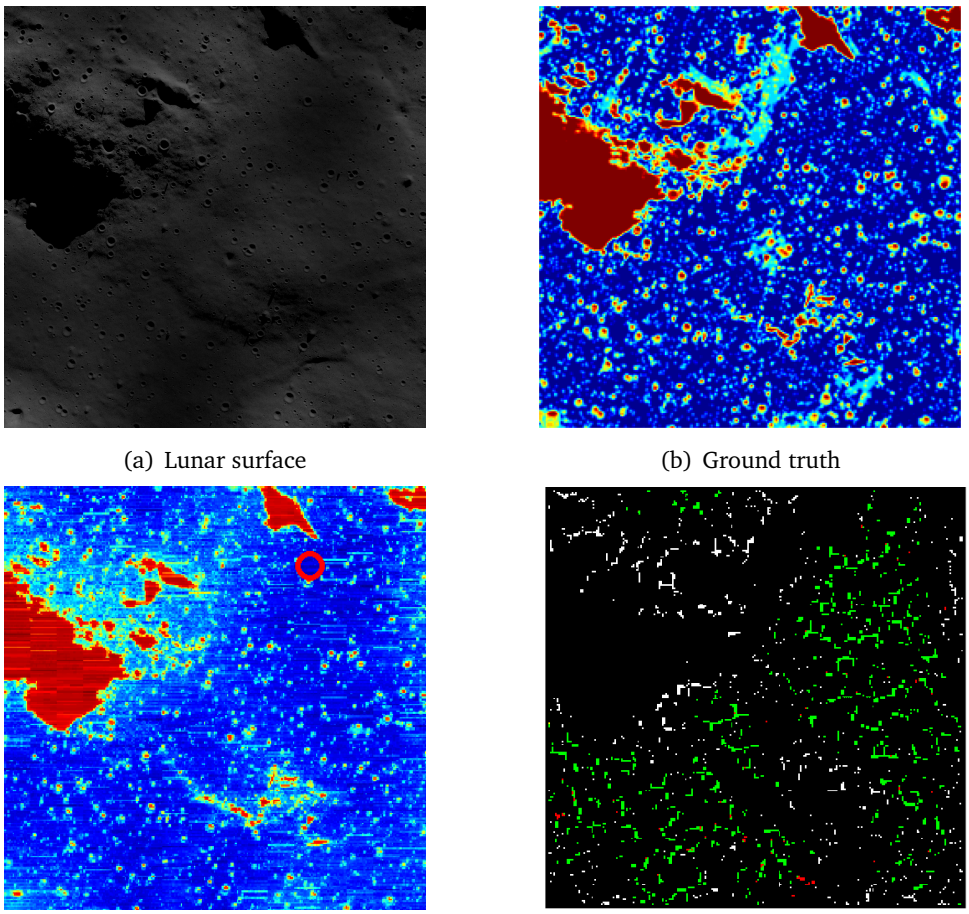


(c) Computed hazard map



(d) Landing Sites Validation

Figure 6.10: Ground truth and computed hazard map comparison, test image 3. In (d), red means False Positive, green True Positive, white False Negative



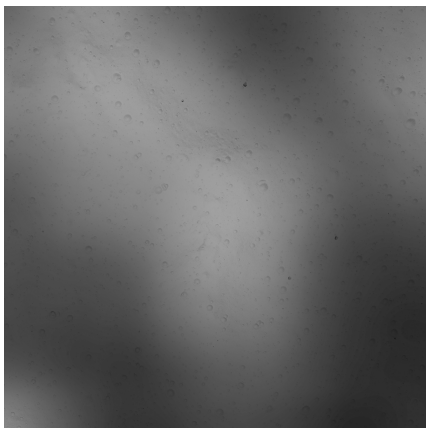
(a) Lunar surface

(b) Ground truth

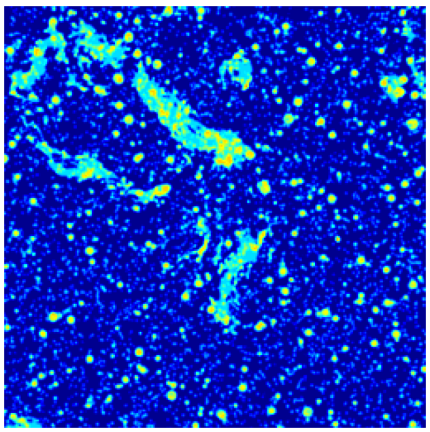
(c) Computed hazard map

(d) Landing Sites Validation

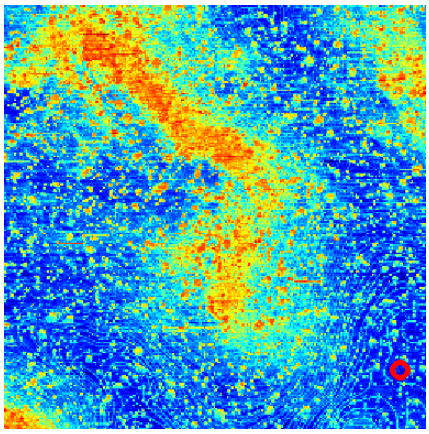
Figure 6.11: Ground truth and computed hazard map comparison, test image 4. In (d), red means False Positive, green True Positive, white False Negative



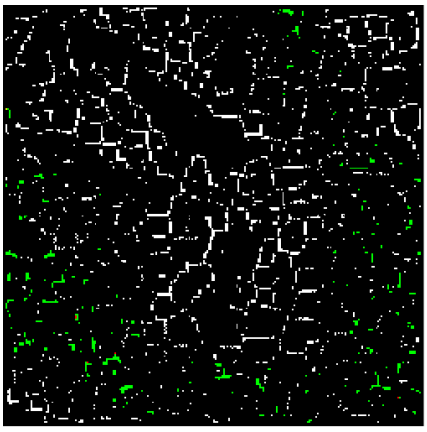
(a) Lunar surface



(b) Ground truth



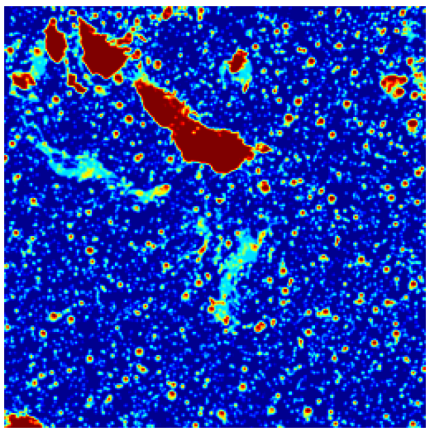
(c) Computed hazard map



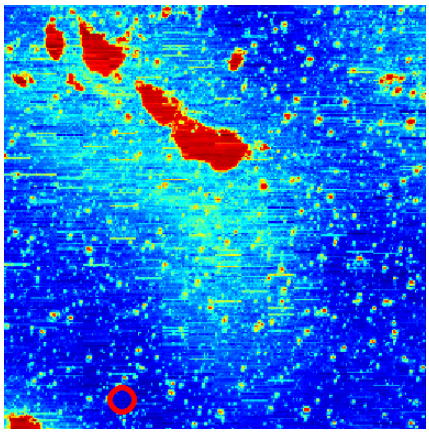
(d) Landing Sites Validation

Figure 6.12: Ground truth and computed hazard map comparison, test image 5. In (d), red means False Positive, green True Positive, white False Negative

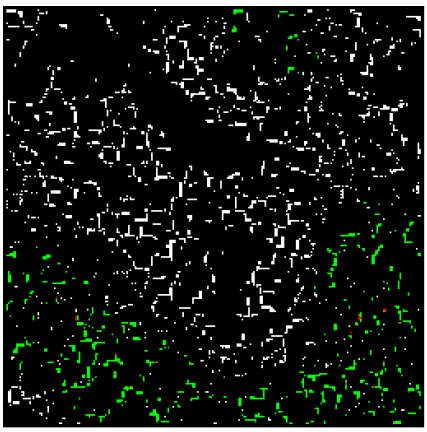
(a) Lunar surface



(b) Ground truth

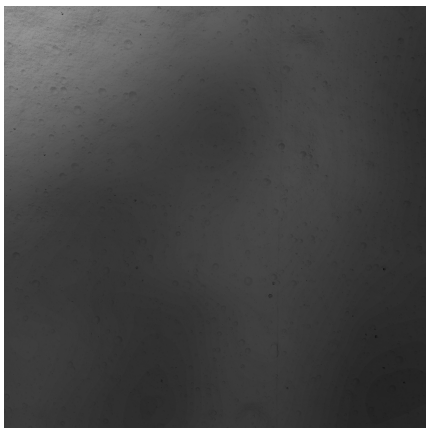


(c) Computed hazard map

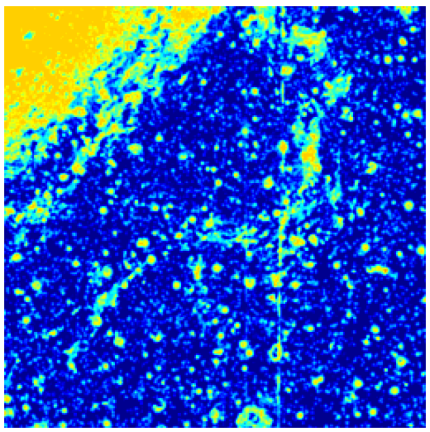


(d) Landing Sites Validation

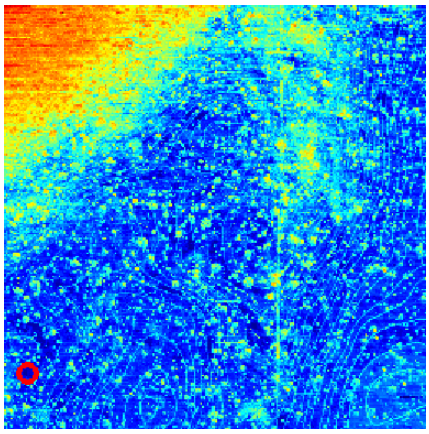
Figure 6.13: Ground truth and computed hazard map comparison, test image 6. In (d), red means False Positive, green True Positive, white False Negative



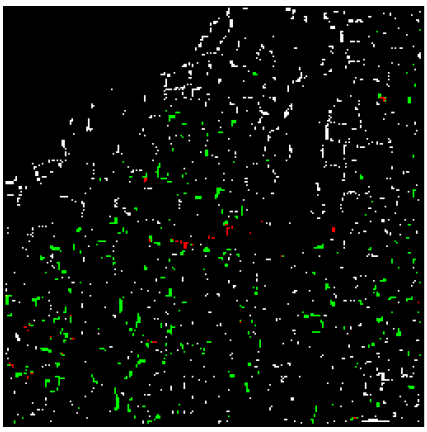
(a) Lunar surface



(b) Ground truth

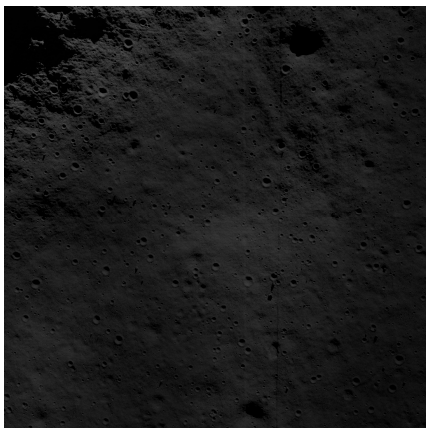


(c) Computed hazard map

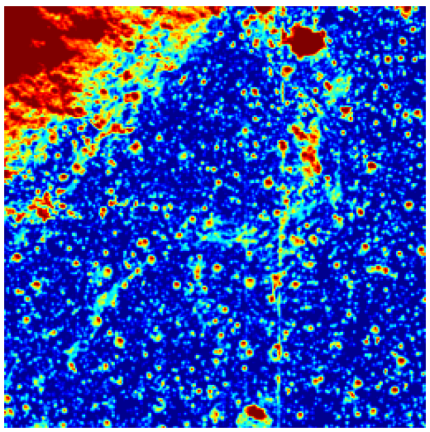


(d) Landing Sites Validation

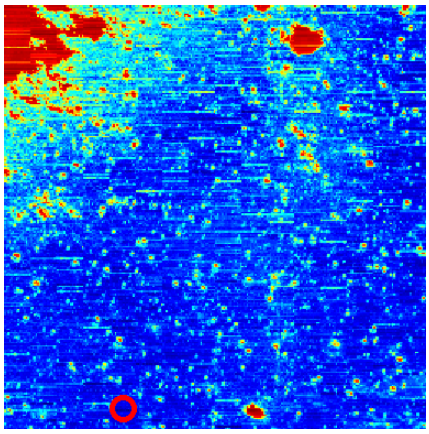
Figure 6.14: Ground truth and computed hazard map comparison, test image 7. In (d), red means False Positive, green True Positive, white False Negative



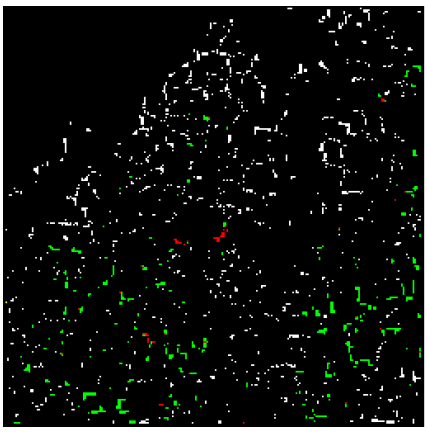
(a) Lunar surface



(b) Ground truth



(c) Computed hazard map



(d) Landing Sites Validation

Figure 6.15: Ground truth and computed hazard map comparison, test image 8. In (d), red means False Positive, green True Positive, white False Negative

6.3.1 Performances removing the blur filter

As stated in Section 5.3, a very slight blur filter is applied after the hazard map has been computed by the artificial neural network in order to relate the near pixels one with the other. Hazard map performances removing the blur filter from the HD system are summarized in Table 6.8: RMSE is increased in average of about 1% both in 15 ° and 80 ° image subsets with respect to the HD with the blur filter. A comparison of the RMSE with or without the blur filter is in Figure 6.16. Landing site candidates identification performances are

Test image	**Sun elevation [°]**	**RMSE [-]**
1	80	0.067585
2	15	0.082091
3	80	0.075522
4	15	0.091924
5	80	0.078826
6	15	0.095384
7	80	0.063034
8	15	0.091215
Mean (Sun 15°)		**0.090154**
Mean (Sun 80°)		**0.071242**
Mean (all)		**0.080698**
Max (all)		**0.095384**
Min (all)		**0.063034**

Table 6.8: Hazard map RMSE removing the blur filter.

listed in Table 6.9: a slight increasing is registered in the low inclination dataset CPI, vice versa in the 80° subset, as it is possible to appreciate in Figure 6.17. Although the overall average results a bit higher for the CPI, the MLV for the high Sun elevation images scored a high 0.93, a value much higher than 0.91 as in the definitive version of the HD system where the blur filter is still present (Fig. 6.18).
For what concerns the first false positive occurrence, it is possible to notice in Table 6.10 and Figure 6.19 that performances are dramatically decreased, due to the lower MLV.
Talking about computational performances, the suppression of the blur filter is negligible. In conclusion, the blur filter is maintained because of the too much increased MLV for the 80° Sun inclination

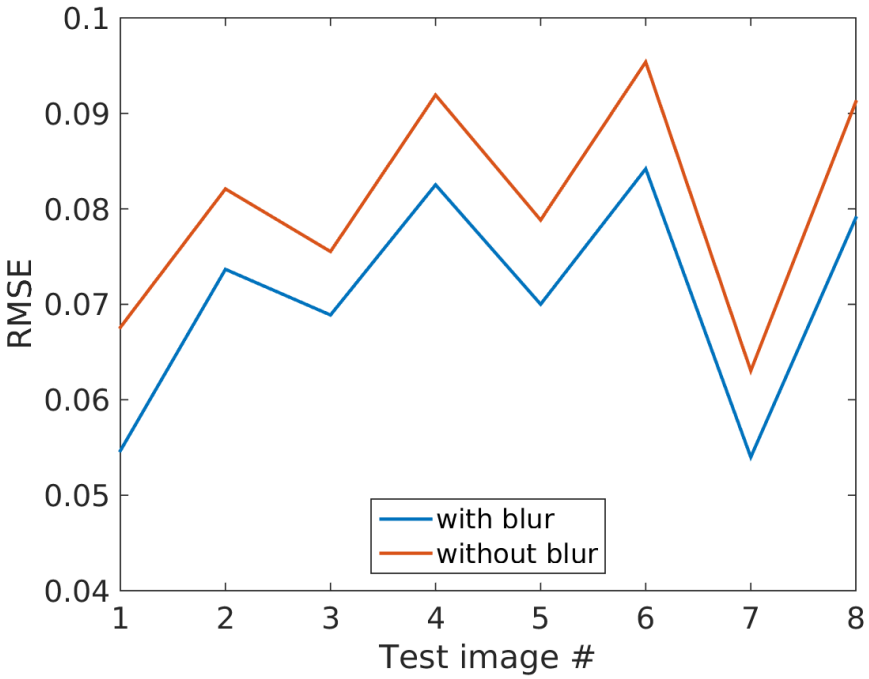


Figure 6.16: Hazard map RMSE of the 8 test images, with or without the blur filter.

Test image	**Sun elevation [°]**	**CPI [-]**	**MLV [-]**
1	80	0.987952	0.973906
2	15	0.960000	0.903421
3	80	0.994012	0.942441
4	15	0.953529	0.689773
5	80	0.990385	0.922346
6	15	0.986635	0.825282
7	80	0.923858	0.877338
8	15	0.933442	0.894146
Mean (Sun 15°)		**0.958401**	**0.828155**
Mean (Sun 80°)		**0.974052**	**0.929008**
Mean (all)		**0.966227**	**0.878582**
Max (all)		**0.994012**	**0.973906**
Min (all)		**0.923858**	**0.689773**

Table 6.9: Hazard detection system performances, results removing the blur filter.

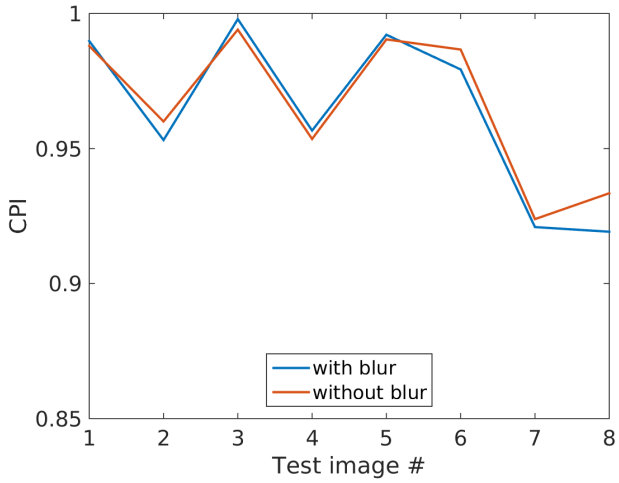


Figure 6.17: CPI of the 8 test images, with or without the blur filter.

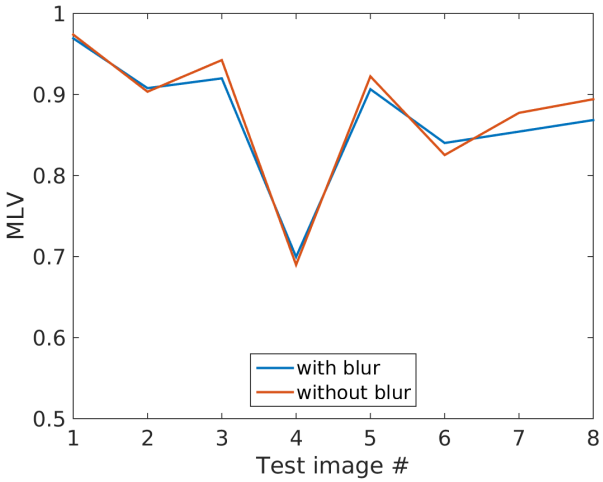


Figure 6.18: MLV of the 8 test images, with or without the blur filter.

test subset, that could mine the hazard detection system capability to find enough landing sites.

Test image	Sun elevation [°]	Position of first FP
1	80	134
2	15	41
3	80	197
4	15	45
5	80	132
6	15	48
7	80	3
8	15	10
Mean (Sun 15°)		116.5
Mean (Sun 80°)		26
Mean (all)		71.25
Max (all)		197
Min (all)		3

Table 6.10: First false positives in the landing site candidates ranking for the test dataset images, results removing the blur filter.

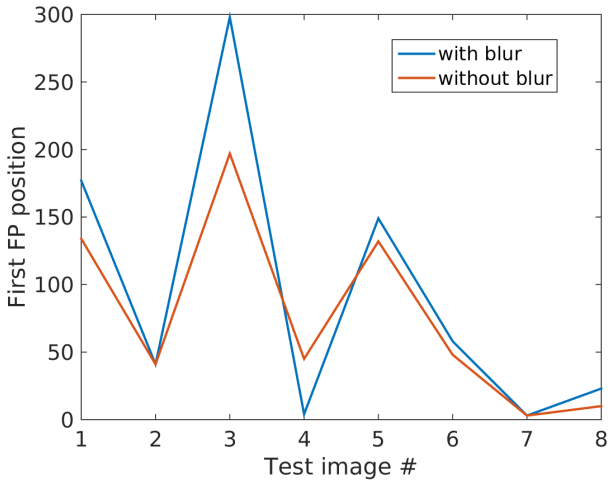


Figure 6.19: First False Positive position in the landing sites ranking for the 8 test images, with or without the blur filter.

6.4 Discarded architectures performances

The same tests done on the hazard detection system are performed on the various alternative architectures presented in Chapter 5, and their results compared. Due to the fact that the various architectures could be also very different in output, it has been proceeded selecting the *minimum computed hazard map threshold for which the HD system yielded at least one landing site per test image*. Hazard threshold precision is set to 10^{-2}. No optimization have been performed on the various discarded architectures because of the great computational effort requested. Therefore, the parameter regarding the first false positive found in the landing site candidates ranking is not taken into account.
Considering the computational efficiency, all the discarded architectures exist only in their Matlab® version, therefore they are not comparable with respect to the profiling done on the final version in Section 6.3. Moreover, development version of the last HD design iteration is written in Python and uses NumPy and OpenCV libraries. Hence, to have a coherent comparison of the various computational costs, a Matlab® version has been developed for the definitive version too. It has been recorded a CPU time of such version of 6.15 s. Computational performances τ will be assessed for every version as a fraction of the time T requested by the very last mentioned Matlab® variant of the final HD system:

$$\tau = \frac{t}{T} \tag{6.4}$$

where t represent the CPU time of the HD version under analysis and $T = 6.15\ s$
Test images 3 and 4 are used as template to able the reader to visualize the various behavior of the surveyed HD version. They represent the same lunar area with two very different Sun inclination angles: 80° and 15°. Such a choice makes the reader aware of the performances with different light conditions, without complicating the presentation with all the test pictures. Quantification of the performances are indeed shown for every element of the test dataset through the RMSE, CPI, MLV parameters defined above in Sections 6.1 and 6.2.
Landing site selection weights are set arbitrarily to 0.5, 0.2, 0.1 respectively for radius, hazard index and distance from the nominal

landing site, although they do not influence the quality parameters used to measure the various versions performances, because they only modify landing site candidates positions in the ranking.

Version A

Hazard map threshold is set to 0.16. τ is 0.18 for a CPU time of 1.10 s. Hence version A is computationally lighter than the definitive HD version.

Hazard map

Performances of the HD architecture in Section 5.6 are depicted in Table 6.11 and a comparison with the definitive version is presented in Figure 6.20. The average RMSE results below 10% and hazard

Test image	**Sun elevation [°]**	**RMSE [-]**
1	80	0.08459
2	15	0.085187
3	80	0.097306
4	15	0.1149
5	80	0.13918
6	15	0.10932
7	80	0.061617
8	15	0.10439
Mean (Sun 15°)		**0.103451**
Mean (Sun 80°)		**0.095673**
Mean (all)		**0.099562**
Max (all)		**0.13918**
Min (all)		**0.061617**

Table 6.11: Hazard map RMSE, version A.

maps computed from images characterized by an higher Sun elevation angle scored a lower error compared to the others. In any case the error difference between the two inclination sets is less than a negligible 1%. This means that version A of the hazard detection system behaves almost independently for what concerns Sun elevation angle and hazard map RMSE. Indeed, the highest similarity between ground truth and computed hazard map belongs to the test image 7, whereas test image 5, whose Sun inclination parameter is 80° too, registered the worst performance of the whole set.

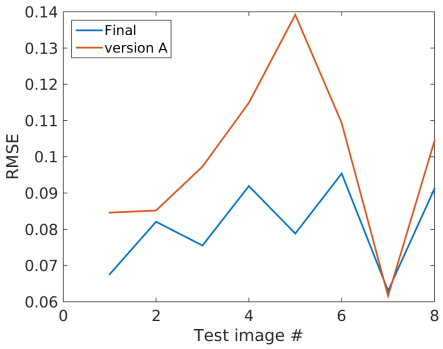
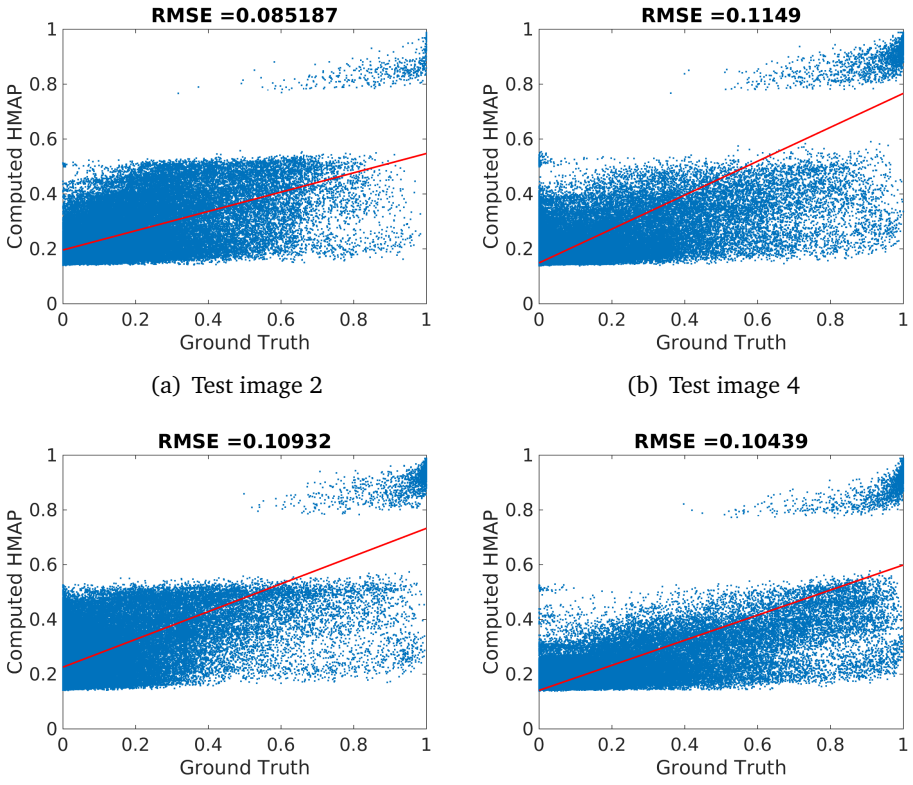


Figure 6.20: RMSE comparison between final HD architecture and version A on the test dataset.



(a) Test image 2

(b) Test image 4

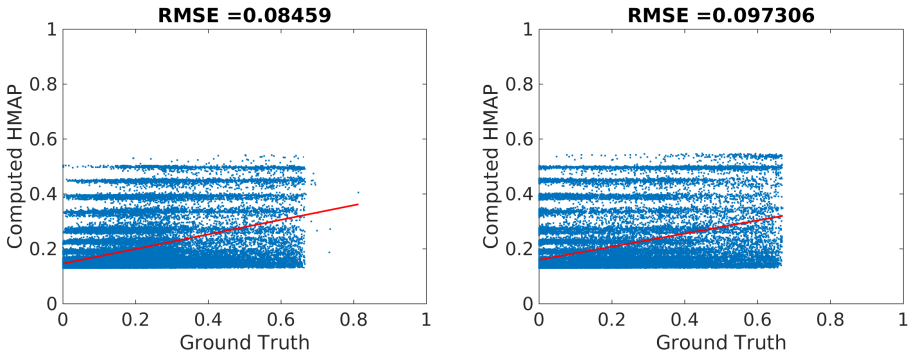
(c) Test image 6

(d) Test image 8

Figure 6.21: RMSE, version A: images with 15° Sun elevation in the test dataset.

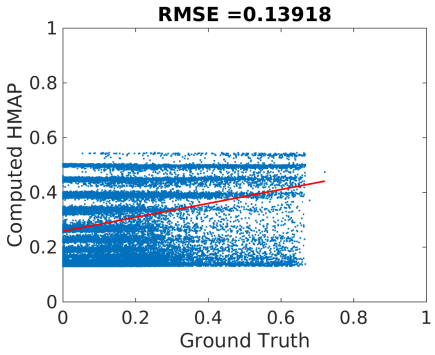
Landing sites

As readable in Table 6.12 and Figure 6.23, parameters are very different between the images with low and high Sun elevation angle.

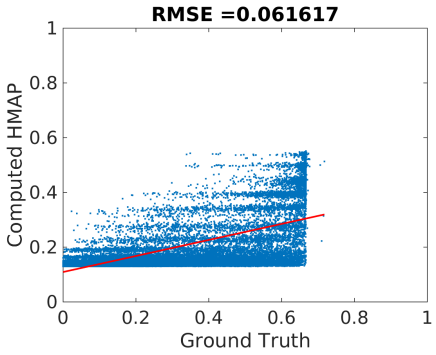


(a) Test image 1

(b) Test image 3

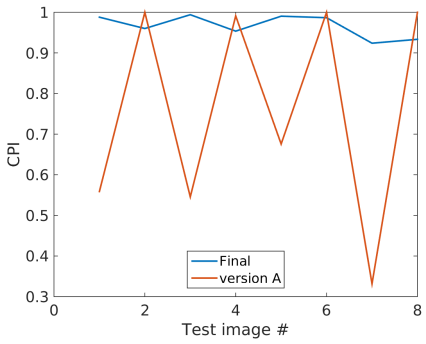


(c) Test image 5

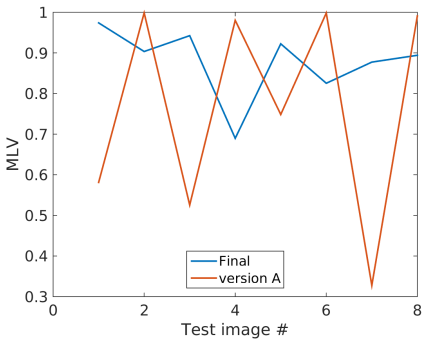


(d) Test image 7

Figure 6.22: RMSE, version A: images with 80° Sun elevation in the test dataset.



(a) CPI



(b) MLV

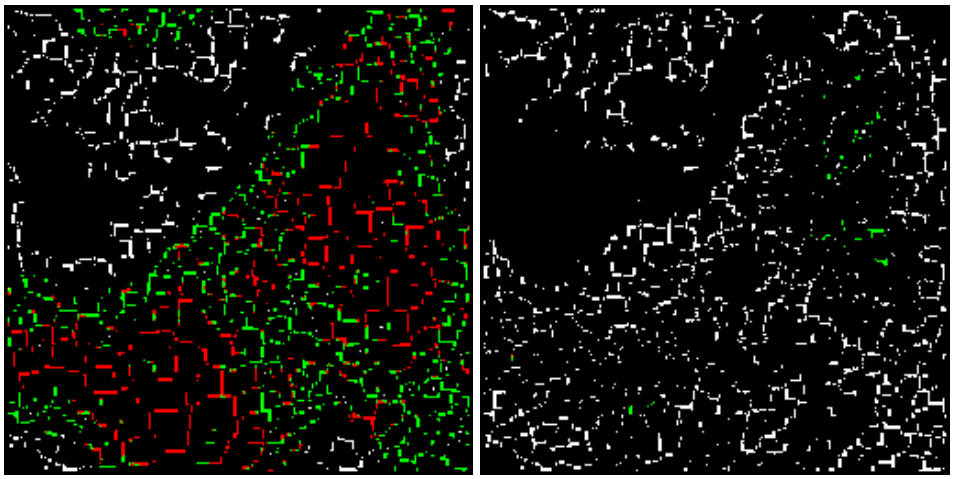
Figure 6.23: Comparison between final HD version and version A in terms of CPI and MLV.

Whereas at 15° the network seems to be very conservative missing almost the totality of the true landing sites (MLV mean over 0.99)

Test image	**Sun elevation [°]**	**CPI [-]**	**MLV [-]**
1	80	0.558791	0.581159
2	15	1.000000	0.998562
3	80	0.545576	0.525463
4	15	0.991071	0.980207
5	80	0.675642	0.748394
6	15	1.000000	0.998261
7	80	0.331575	0.327474
8	15	1.000000	0.990871
Mean (Sun 15°)		**0.997768**	**0.991975**
Mean (Sun 80°)		**0.527896**	**0.545623**
Mean (all)		**0.762832**	**0.768799**
Max (all)		**1.000000**	**0.998562**
Min (all)		**0.331575**	**0.327474**

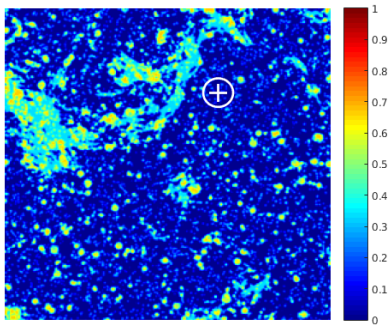
Table 6.12: Hazard detection system performances, version A.

and yielding the 99.78% of true positives in average, at 80° both values fall down to almost 50%. In case of CPI, such a value is not acceptable, because it states that the target landing site has the same probability to be a true positive or a false positive. Assessing a total CPI average of 0.76 the version A seems to be limited to operations with low Sun inclination angles, and therefore it falls out of the requirements for the hazard detection system studied in this thesis.

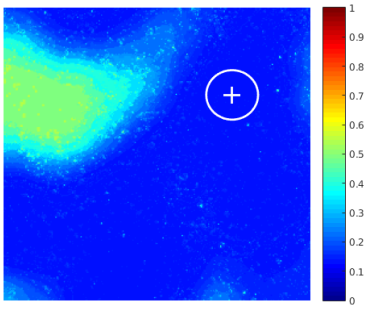


(a) Landing Sites, test image 3 (b) Landing Sites, test image 4

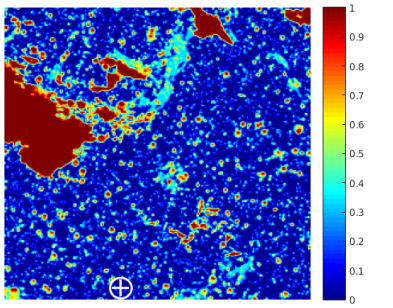
Figure 6.24: Resulting landing sites on test images 3 and 4. Green represents a True Positive, red a False Positive, white a False Negative.



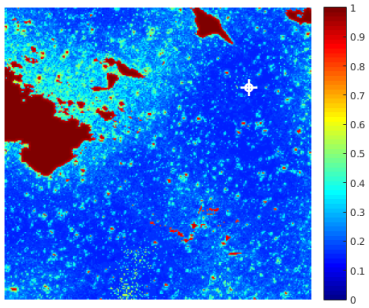
(a) True hazard map, test image 3



(b) Computed hazard map, test image 3



(c) True hazard map, test image 4



(d) Computed hazard map, test image 4

Figure 6.25: Comparison between ground truth and HD version A hazard map.

Version B

Hazard map threshold is set to 0.10.
τ is 13.42 for a CPU time of 82.52 s. Thus version B is sensibly computationally heavier than the definitive HD system.

Hazard map

Hazard map RMSE of the HD architecture in Section 5.6 is depicted in Table 6.13 and a comparison with the definitive version is presented in Figure 6.26.

Test image	**Sun elevation [$^\circ$]**	**RMSE [-]**
1	80	0.028041
2	15	0.11745
3	80	0.027092
4	15	0.12489
5	80	0.029482
6	15	0.14058
7	80	0.028601
8	15	0.12088
Mean (Sun 15°)		**0.028304**
Mean (Sun 80°)		**0.12595**
Mean (all)		**0.077127**
Max (all)		**0.14058**
Min (all)		**0.027092**

Table 6.13: Hazard map RMSE, version B.

Average RMSE for the whole test set remains lower than 8%. Although Sun elevation angle is supplied to the network, Table 6.13 shows a drastic difference among images with low and high inclination angle: the 80° set scored a very low 2.8% average error with respect to the ground truth, whereas the 15° set increases the RMSE mean up to 12.6%. Recalling the definition of RMSE (Equation 6.1), it represents the data dispersion about the regression line. In Figures 6.27 and 6.28 it is possible to notice that even though 80° test subset registered a much lower RMSE, corresponding regression lines are characterized by a very low angular coefficient (Fig. 6.28). Therefore, as the hazard threshold grows, the quality of the computed haz-

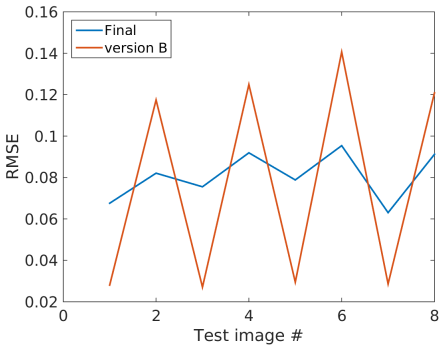
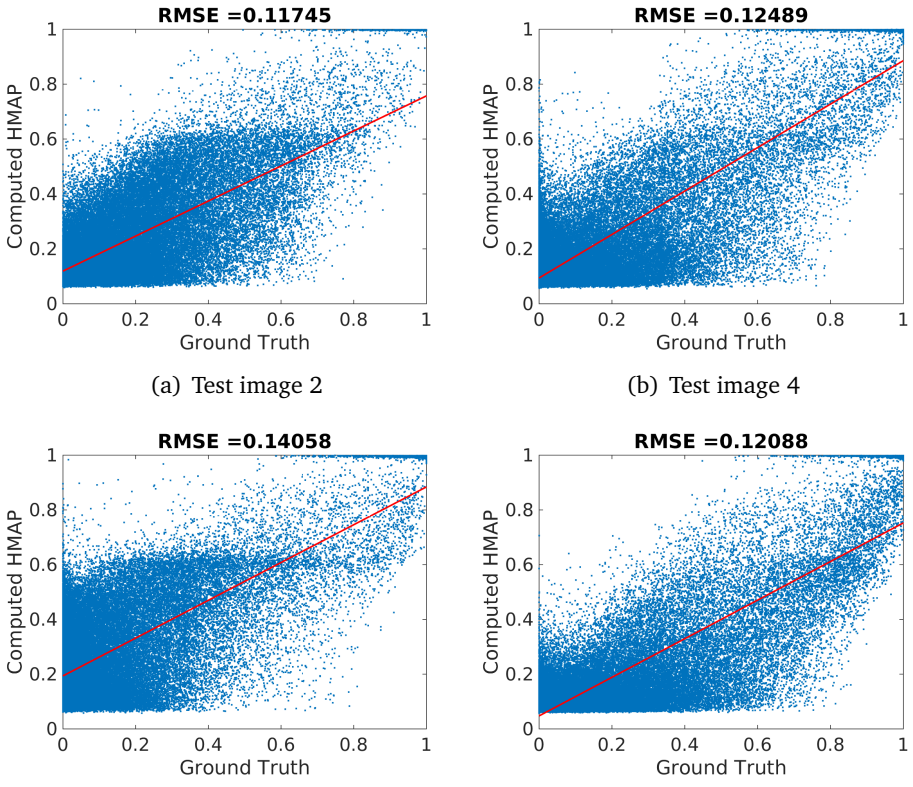


Figure 6.26: RMSE comparison between final HD architecture and version B on the test dataset.

ard map decreases. This behavior may suggest the use of an hazard index threshold adaptable to the current Sun inclination angle.



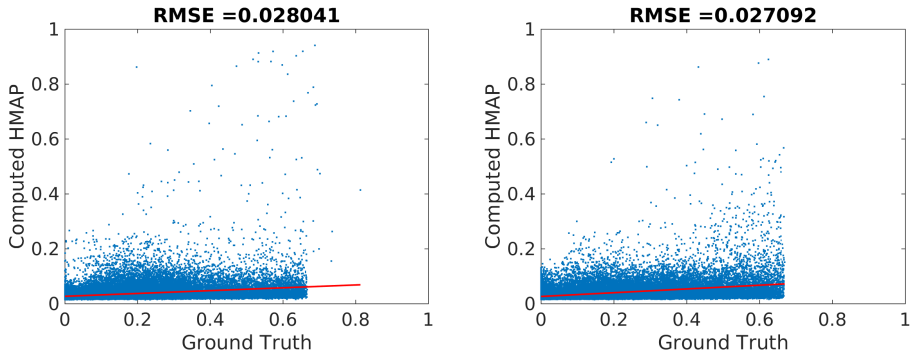
(a) Test image 2

(b) Test image 4

(c) Test image 6

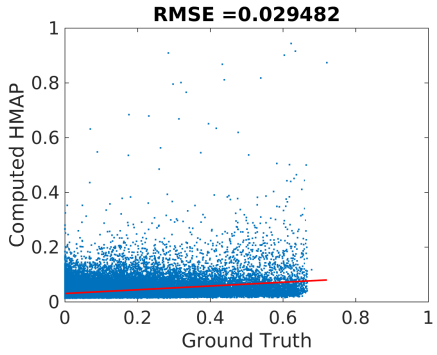
(d) Test image 8

Figure 6.27: RMSE, version B: images with 15° Sun elevation in the test dataset.

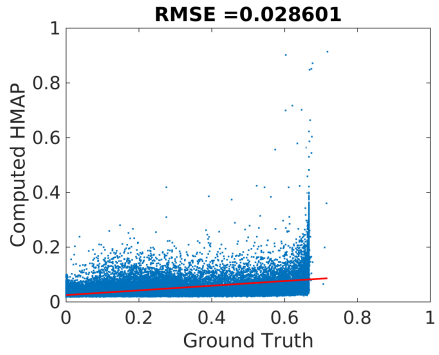


(a) Test image 1

(b) Test image 3



(c) Test image 5



(d) Test image 7

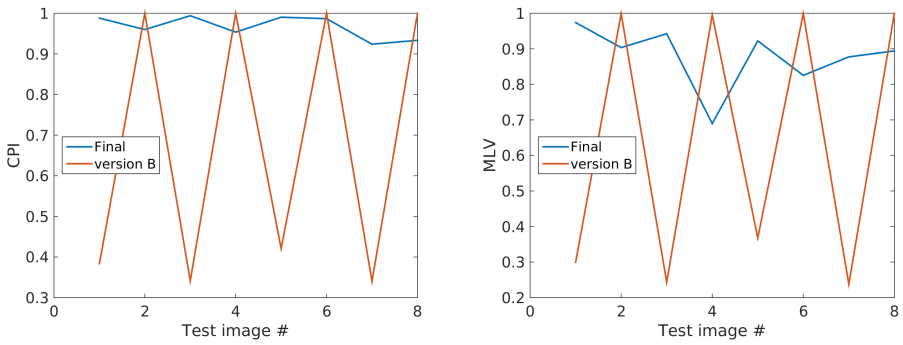
Figure 6.28: RMSE, version B: images with 80° Sun elevation in the test dataset.

Landing sites

Landing sites computation performances are depicted in Table 6.14, while in Figure 6.29 a comparison with the final HD system is presented. The extreme different behavior of the HD version B with respect to the two Sun inclination angle subsets is confirmed taking into account the landing site candidates computed. In 15° images no false positives are found, with a perfect CPI=1 in all the four cases. On the other hand, the low hazard threshold adopted makes the HD to miss the majority of landing sites present in the ground truth: the average MLV for this Sun inclination test subset surpasses 0.99. Performances on high Sun elevation images are quite poor: the best CPI is recorded in test image 5 with a mere 0.42, meaning that more than half of the computed landing site candidates are false positives. MLV is sensibly lower in this case, with an average of 0.29. These values are easily understandable looking at Figures 6.30 and 6.31,

Test image	Sun elevation [°]	CPI [-]	MLV [-]
1	80	0.383553	0.299453
2	15	1.000000	0.998921
3	80	0.341503	0.243744
4	15	1.000000	0.995294
5	80	0.420718	0.367892
6	15	1.000000	0.999367
7	80	0.340317	0.237552
8	15	1.000000	0.999400
Mean (Sun 15°)		**1.00000**	**0.998246**
Mean (Sun 80°)		**0.371523**	**0.287160**
Mean (all)		**0.685761**	**0.642703**
Max (all)		**1.00000**	**0.999400**
Min (all)		**0.340317**	**0.237552**

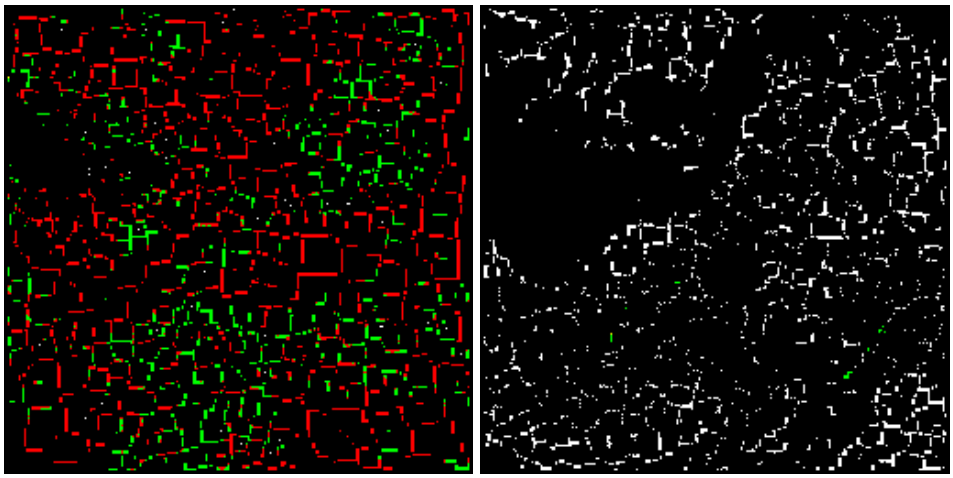
Table 6.14: Hazard detection system performances, version B.



(a) CPI (b) MLV

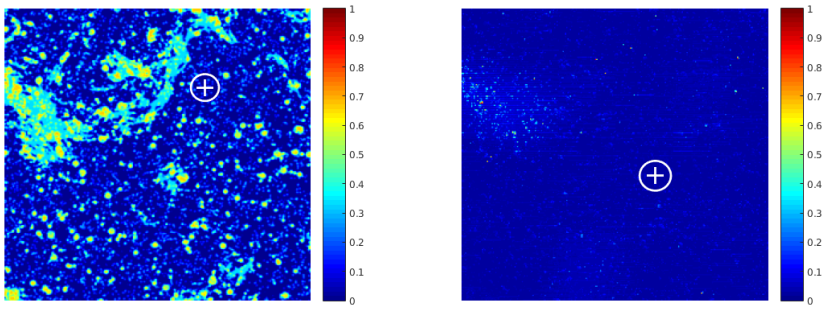
Figure 6.29: Comparison between final HD version and version B in terms of CPI and MLV.

where the 80° computed hazard map is almost completely missing hazardous areas.
This version, being limited to operations with low Sun elevation angles only, does not satisfy system requirements.

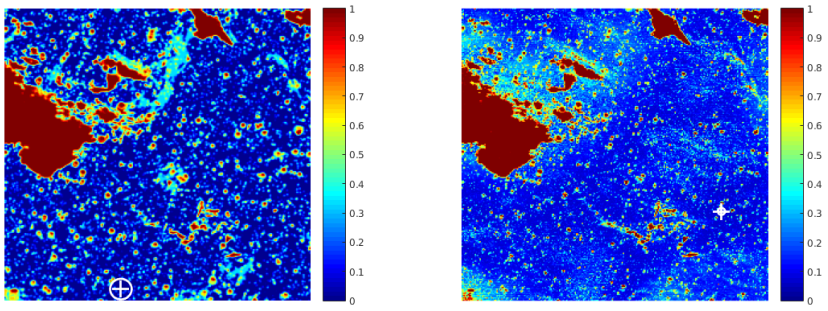


(a) Landing Sites, test image 3 (b) Landing Sites, test image 4

Figure 6.30: Resulting landing sites on test images 3 and 4. Green represents a True Positive, red a False Positive, white a False Negative.



(a) True hazard map, test image 3 (b) Computed hazard map, test image 3



(c) True hazard map, test image 4 (d) Computed hazard map, test image 4

Figure 6.31: Comparison between ground truth and HD version B hazard map.

Version C

Hazard map threshold is set to 0.15.
τ is 23.77 for a CPU time of 146.19 s. Thus version C is computationally heavier than the definitive HD system.

Hazard map

Hazard map RMSE of the HD architecture in Section 5.6 is depicted in Table 6.15 and a comparison with the definitive version is presented in Figure 6.32. Also in HD version C RMSE is lower for 80°

Test image	**Sun elevation [°]**	**RMSE [-]**
1	80	0.06788
2	15	0.11941
3	80	0.094068
4	15	0.12413
5	80	0.12163
6	15	0.14419
7	80	0.064186
8	15	0.12778
Mean (Sun 15°)		**0.12888**
Mean (Sun 80°)		**0.086941**
Mean (all)		**0.10791**
Max (all)		**0.14419**
Min (all)		**0.064186**

Table 6.15: Hazard map RMSE, version C.

Sun inclination angle test subset. As previous versions, 15° subset has a higher angular coefficient of the regression line, as depicted in Figures 6.33 and 6.34. Difference in RMSE is not particularly significant, assessing the average between the two inclination subsets of about 3%.

Landing sites

Landing sites computation performances are depicted in Table 6.16, while in Figure 6.35 a comparison with the final HD system is presented. In this HD version it is possible to notice an improvement in

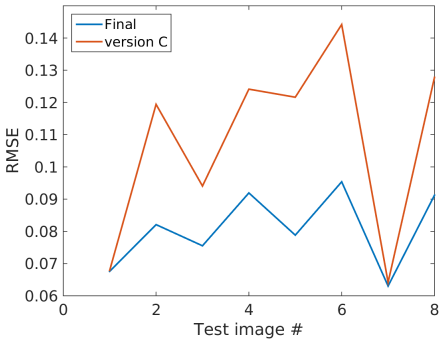
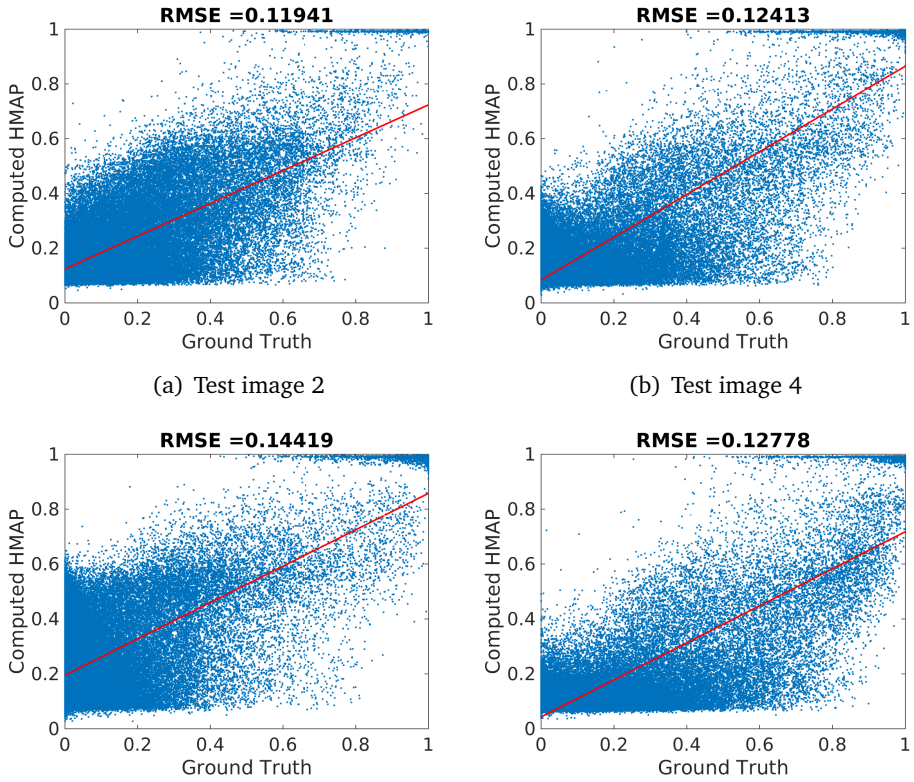


Figure 6.32: RMSE comparison between final HD architecture and version A on the test dataset.



(a) Test image 2

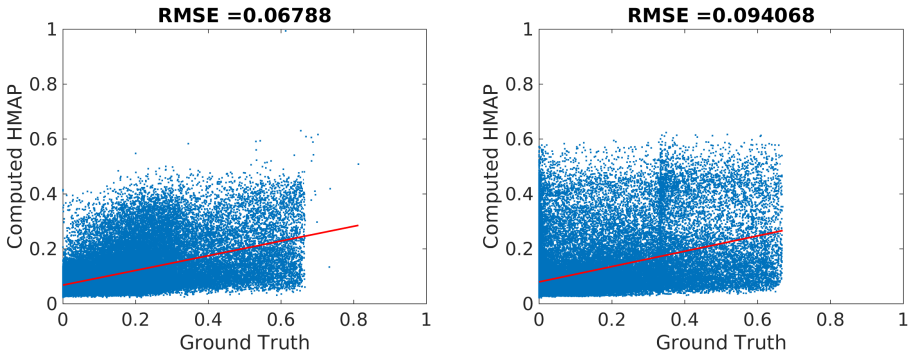
(b) Test image 4

(c) Test image 6

(d) Test image 8

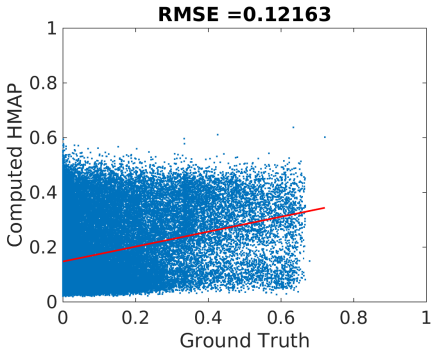
Figure 6.33: RMSE, version C: images with 15° Sun elevation in the test dataset.

the identification of correct landing sites with the 80° subset, with a CPI=0.69, means that more than 2 candidates out of three are correctly understood. With the low inclination subset the network as-

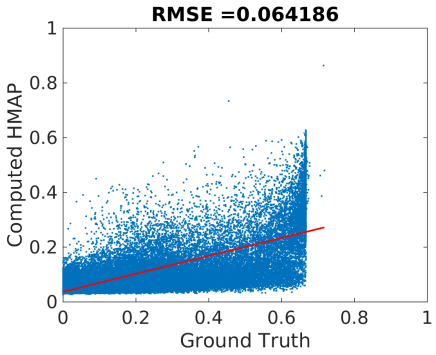


(a) Test image 1

(b) Test image 3

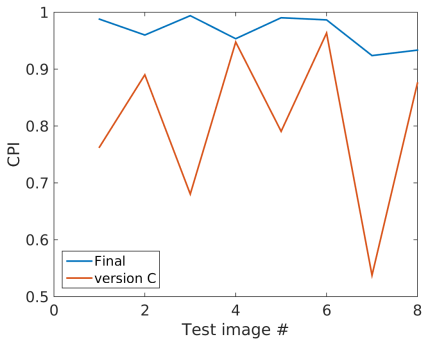


(c) Test image 5

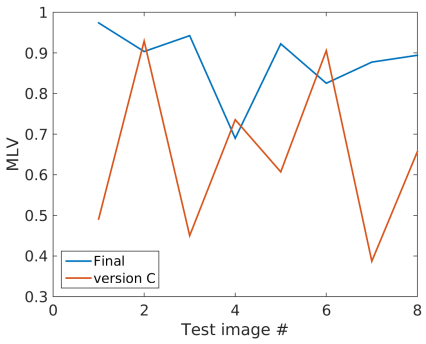


(d) Test image 7

Figure 6.34: RMSE, version C: images with 80° Sun elevation in the test dataset.



(a) CPI



(b) MLV

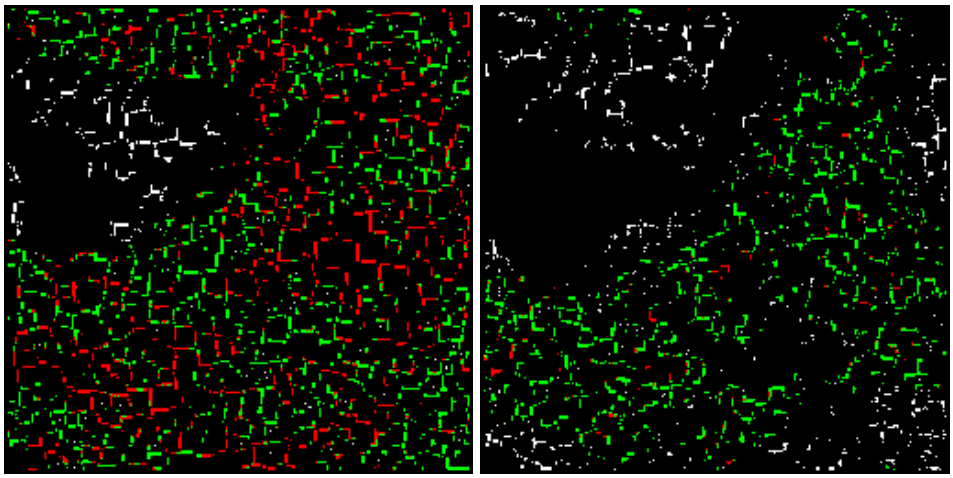
Figure 6.35: Comparison between final HD version and version C in terms of CPI and MLV.

sesses a good CPI=0.92. Total CPI average is the best recorded up to now between the discarded versions: 0.81.

Test image	**Sun elevation [°]**	**CPI [-]**	**MLV [-]**
1	80	0.762776	0.491012
2	15	0.890086	0.929196
3	80	0.680342	0.450205
4	15	0.947624	0.735573
5	80	0.790599	0.607094
6	15	0.963470	0.905593
7	80	0.537498	0.387166
8	15	0.875456	0.656697
Mean (Sun 15°)		**0.919159**	**0.806765**
Mean (Sun °)		**0.692804**	**0.483869**
Mean (all)		**0.805982**	**0.645317**
Max (all)		**0.963470**	**0.929196**
Min (all)		**0.537498**	**0.387166**

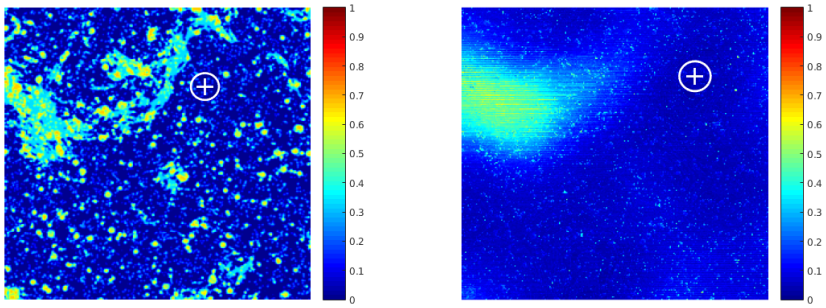
Table 6.16: Hazard detection system performances, version C.

MLV for the low inclination subset is decreased to 0.81, whereas for the 80° test images is 0.48. The general increase in the overall performance of this architecture with respect to versions A and B can be also observed qualitatively in Figure 6.37, where the most hazardous area of the high Sun elevation test image 3 in the upper-left of the frame is identified, even if without great accuracy. Small scale dangerous zones are still almost no identified for the same test image.

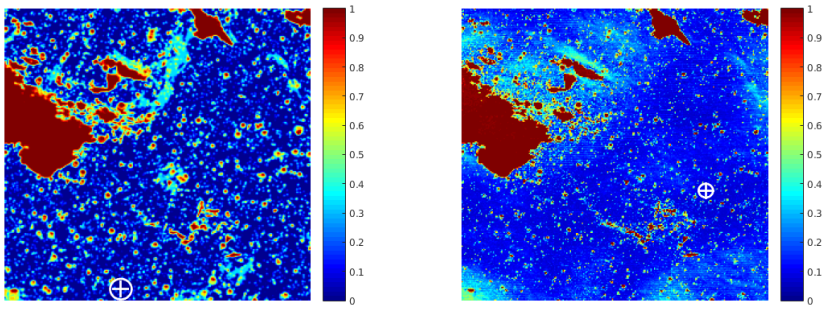


(a) Landing Sites, test image 3 (b) Landing Sites, test image 4

Figure 6.36: Resulting landing sites on test images 3 and 4. Green represents a True Positive, red a False Positive, white a False Negative.



(a) True hazard map, test image 3 (b) Computed hazard map, test image 3



(c) True hazard map, test image 4 (d) Computed hazard map, test image 4

Figure 6.37: Comparison between ground truth and HD version C hazard map.

Version D

Hazard map threshold: 0.12.
τ is 0.30 for a CPU time of 1.87 s. Thus version D is computationally lighter than the definitive HD system.

Hazard map

Hazard map RMSE of the HD architecture in Section 5.6 is depicted in Table 6.17 and a comparison with the definitive version is presented in Figure 6.38. Version D performs terribly bad in both low and high

Test image	**Sun elevation [°]**	**RMSE [-]**
1	80	0.093038
2	15	0.2131
3	80	0.094844
4	15	0.2815
5	80	0.092447
6	15	0.26911
7	80	0.096226
8	15	0.23252
Mean (Sun 15°)		**0.249058**
Mean (Sun 80°)		**0.094139**
Mean (all)		**0.1716**
Max (all)		**0.2815**
Min (all)		**0.092447**

Table 6.17: Hazard map RMSE, version D.

Sun inclination angle test images. A part from the worst total average registered up to now in the RMSE value, from Figures 6.39 and 6.40 the regression line between ground truth and computed hazard maps predicts a completely wrong comprehension of the terrain features by the ANN. With 15° images angular coefficient of the regression line is even negative. Moreover, the fact that more or less every regression line in the test set is horizontal (about null angular coefficient), means that the ANN interprets the input as it were a flat surface, without understanding any of the terrain features.

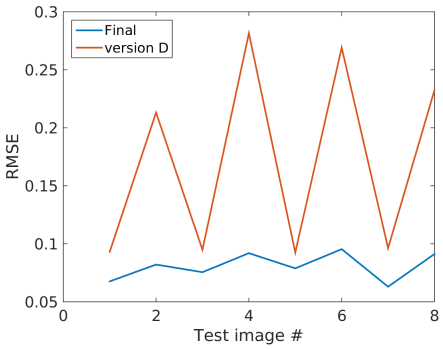
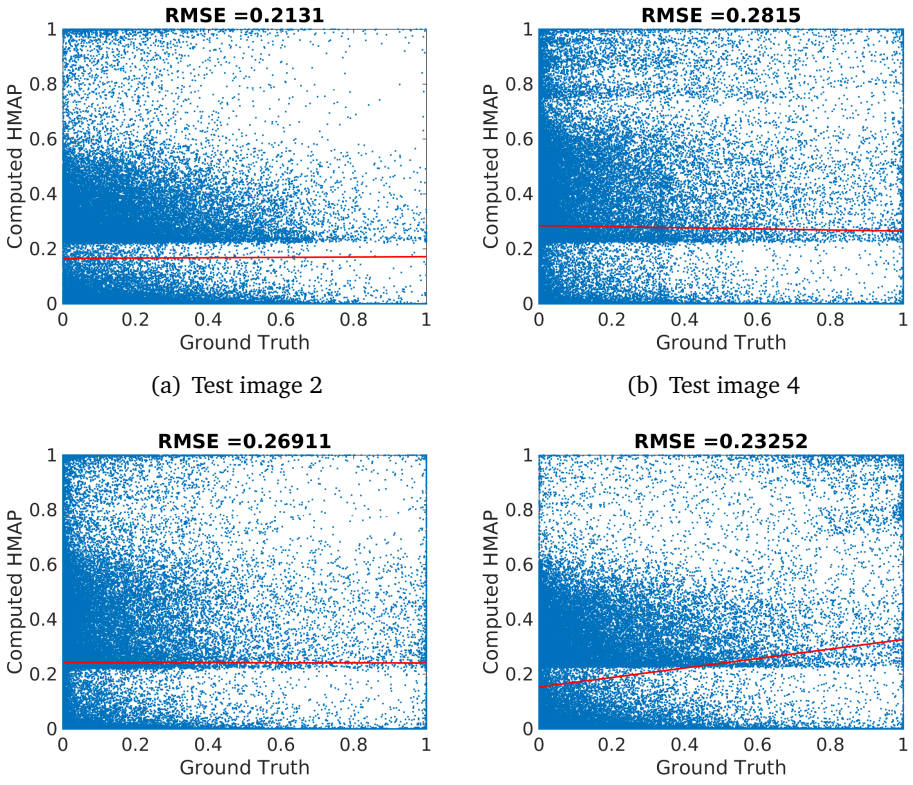


Figure 6.38: RMSE comparison between final HD architecture and version D on the test dataset.



(a) Test image 2

(b) Test image 4

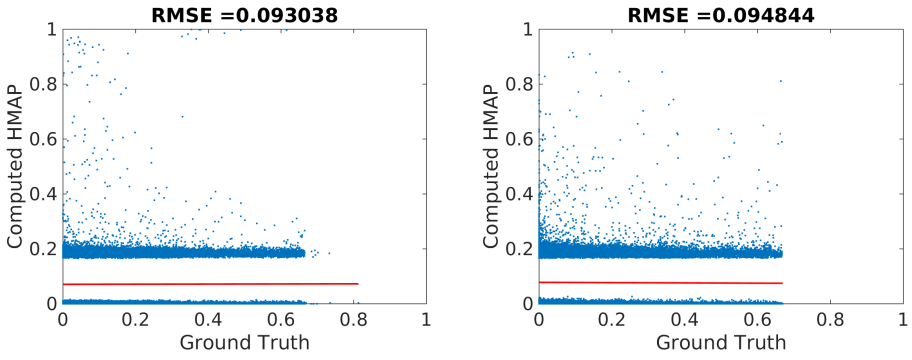
(c) Test image 6

(d) Test image 8

Figure 6.39: RMSE, version D: images with 15° Sun elevation in the test dataset.

Landing sites

Landing sites computation performances are depicted in Table 6.18, while in Figure 6.41 a comparison with the final HD system is pre-

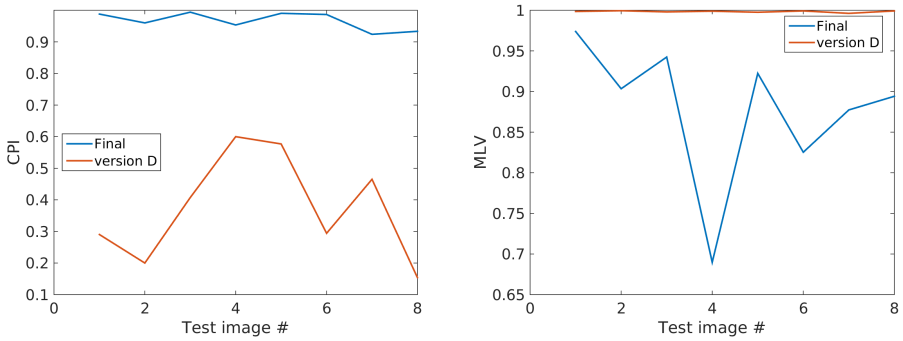


(a) Test image 1 (b) Test image 3

(c) Test image 5 (d) Test image 7

Figure 6.40: RMSE, version D: images with 80° Sun elevation in the test dataset.

sented. A very poor performance is recorded in landing sites iden-



(a) CPI (b) MLV

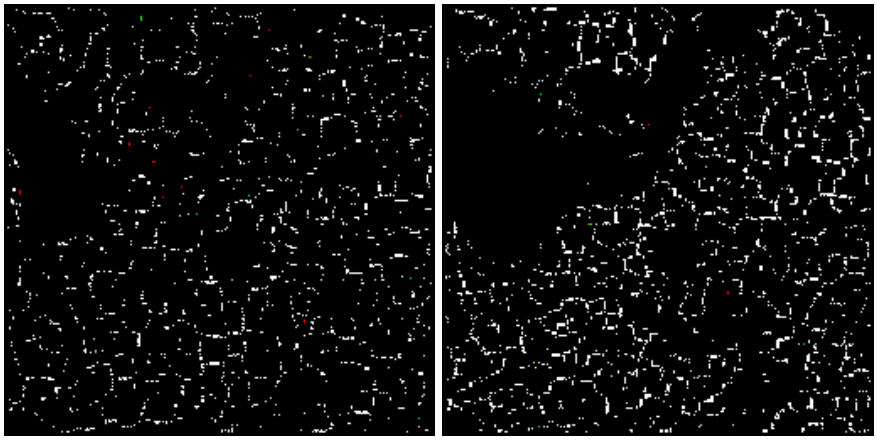
Figure 6.41: Comparison between final HD version and version D in terms of CPI and MLV.

tification too: CPI average stops at 0.37, while MLV is over 0.99 for

Test image	Sun elevation [°]	CPI [-]	MLV [-]
1	80	0.290323	0.998534
2	15	0.200000	0.999460
3	80	0.407407	0.997983
4	15	0.600000	0.998910
5	80	0.576923	0.997559
6	15	0.294118	0.999209
7	80	0.465116	0.996207
8	15	0.153846	0.999200
Mean (Sun 15°)		**0.311991**	**0.999195**
Mean (Sun 80°)		**0.434942**	**0.997571**
Mean (all)		**0.373467**	**0.998383**
Max (all)		**0.600000**	**0.999460**
Min (all)		**0.153846**	**0.996207**

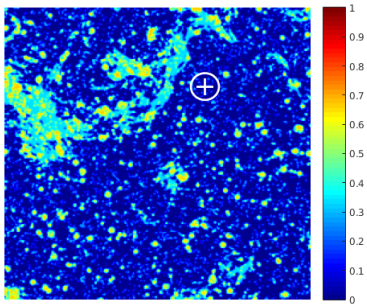
Table 6.18: Hazard detection system performances, version D.

every image in the test dataset. In Figures 6.43 it is possible to notice that version D architecture is unable to completely understand even large shadows, usually the easiest feature to detect for the neural network.
This HD version does not fulfill the system requirements.

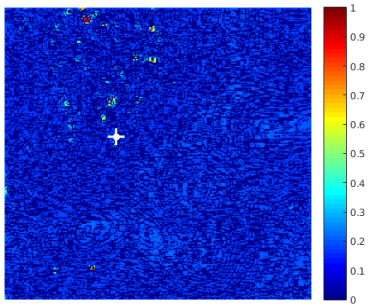


(a) Landing Sites, test image 3 (b) Landing Sites, test image 4

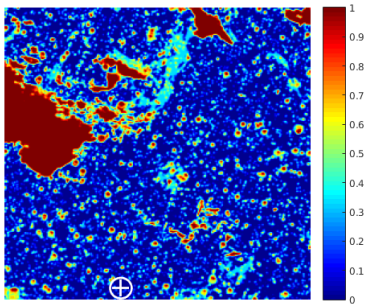
Figure 6.42: Resulting landing sites on test images 3 and 4. Green represents a True Positive, red a False Positive, white a False Negative.



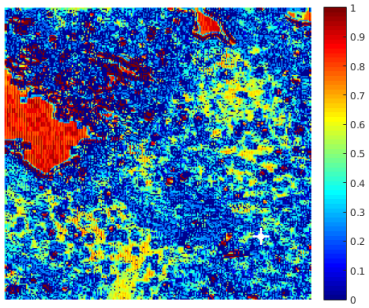
(a) True hazard map, test image 3



(b) Computed hazard map, test image 3



(c) True hazard map, test image 4



(d) Computed hazard map, test image 4

Figure 6.43: Comparison between ground truth and HD version D hazard map.

Version E

Hazard map threshold: 0.08.
τ is 0.31 for a CPU time of 1.90 s. Thus version E is computationally lighter than the definitive HD system.

Hazard map

Hazard map RMSE of the HD architecture in Section 5.6 is depicted in Table 6.19 and a comparison with the definitive version is presented in Figure 6.44. RMSE performances looks better than version D:

Test image	**Sun elevation [$^\circ$]**	**RMSE [-]**
1	80	0.08988
2	15	0.12129
3	80	0.11232
4	15	0.13375
5	80	0.14111
6	15	0.16121
7	80	0.070602
8	15	0.13049
Mean (Sun 15°)		**0.13669**
Mean (Sun 80°)		**0.10348**
Mean (all)		**0.12008**
Max (all)		**0.16121**
Min (all)		**0.070602**

Table 6.19: Hazard map RMSE, version E.

an average error of 12% is computed among all the test images. In Figures 6.45 and 6.46, angular coefficient of the various regression lines seems much more similar to a theoretical perfect result of a bisecting line. Even in this version E of the HD system, RMSE is lower for high Sun elevation images, but regression line coefficient is higher instead in the 15° subset.

Landing sites

Landing sites computation performances are depicted in Table 6.20, while in Figure 6.47 a comparison with the final HD system is pre-

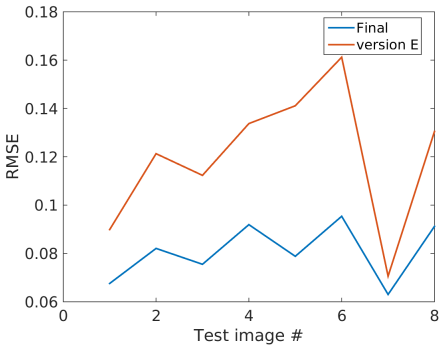
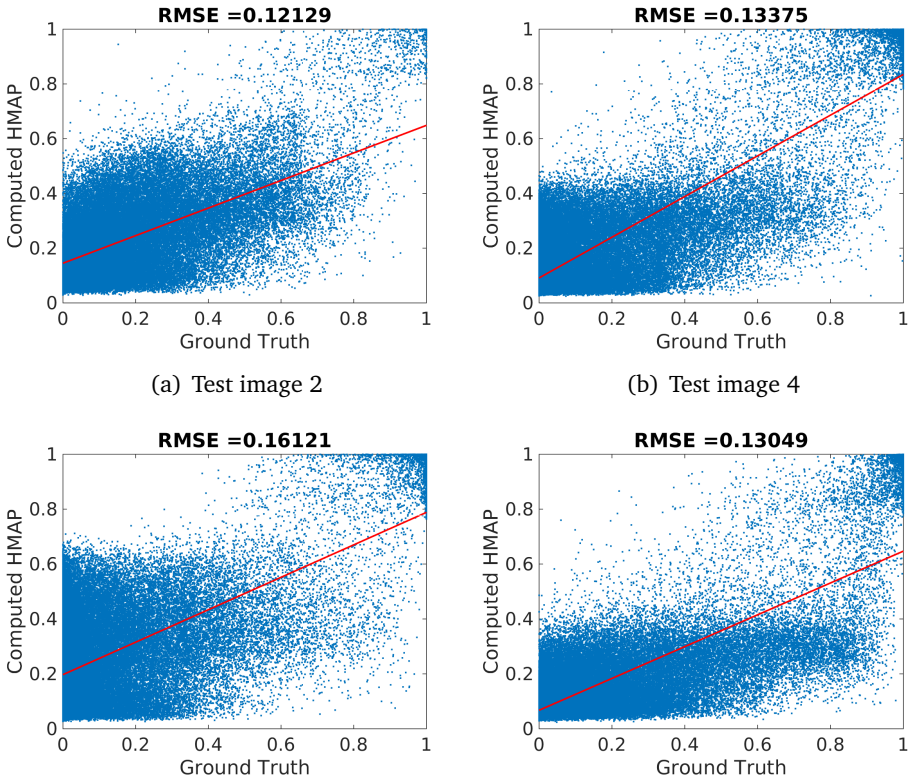


Figure 6.44: RMSE comparison between final HD architecture and version E on the test dataset.



(a) Test image 2

(b) Test image 4

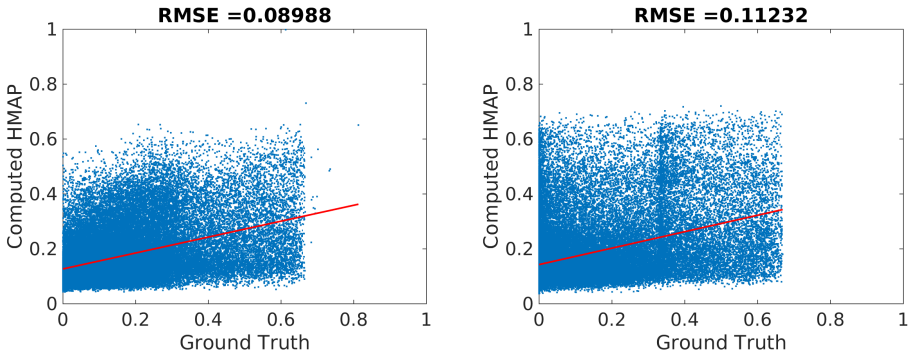
(c) Test image 6

(d) Test image 8

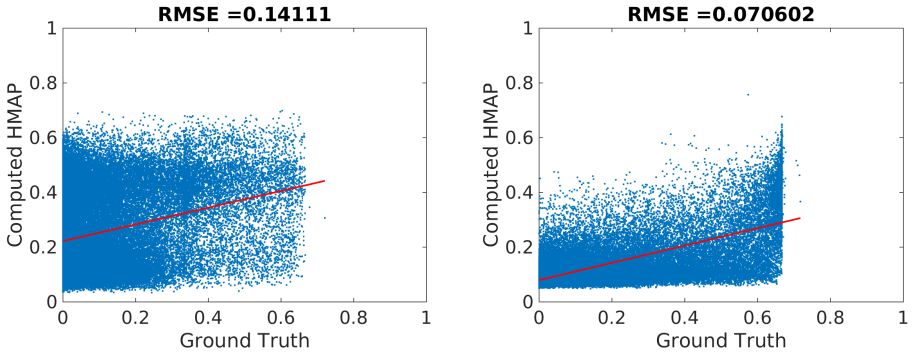
Figure 6.45: RMSE, version E: images with 15° Sun elevation in the test dataset.

sented.

Version E holds the best CPI result among all the discarded versions up to now for the average among all the images: 0.88, with an

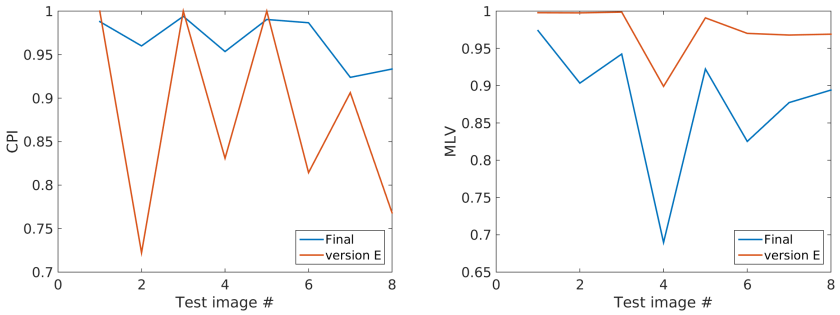


(a) Test image 1 (b) Test image 3



(c) Test image 5 (d) Test image 7

Figure 6.46: RMSE, version E: images with 80° Sun elevation in the test dataset.



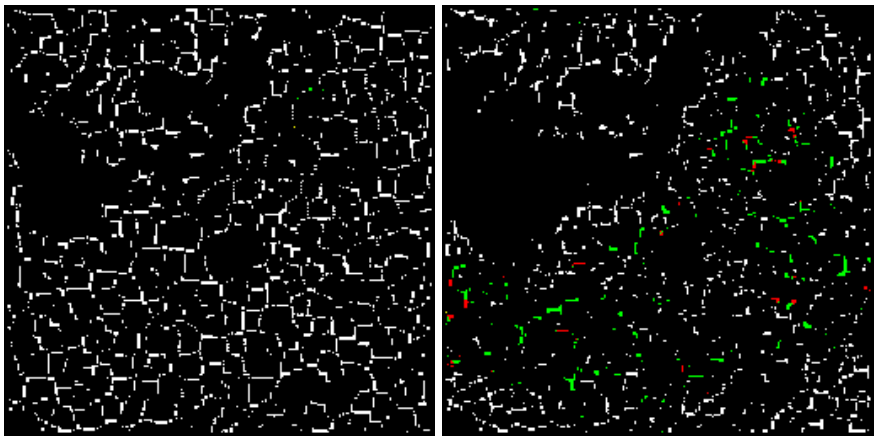
(a) CPI (b) MLV

Figure 6.47: Comparison between final HD version and version E in terms of CPI and MLV.

amazing 0.98 for the high Sun inclination subset. MLV is quite high due to the hazard index of 0.08, the lowest among all the HD architecture variants. It is possible to spot in Figures 6.49 the effects of

Test image	Sun elevation [°]	CPI [-]	MLV [-]
1	80	1.000000	0.997884
2	15	0.722222	0.997666
3	80	1.000000	0.998715
4	15	0.830832	0.899072
5	80	1.000000	0.990947
6	15	0.814346	0.970138
7	80	0.906250	0.967914
8	15	0.768116	0.969078
Mean (Sun 15°)		**0.783879**	**0.958988**
Mean (Sun 80°)		**0.976562**	**0.988865**
Mean (all)		**0.880221**	**0.973927**
Max (all)		**1.00000**	**0.998715**
Min (all)		**0.768116**	**0.899072**

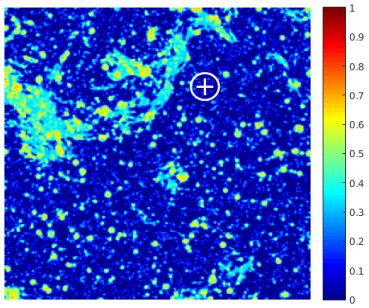
Table 6.20: Hazard detection system performances, version E.



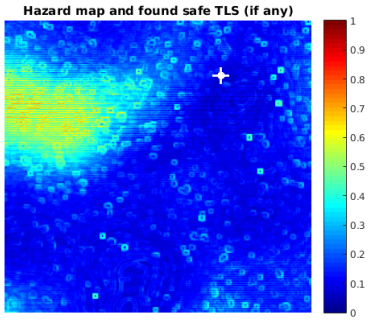
(a) Landing Sites, test image 3 (b) Landing Sites, test image 4

Figure 6.48: Resulting landing sites on test images 3 and 4. Green represents a True Positive, red a False Positive, white a False Negative.

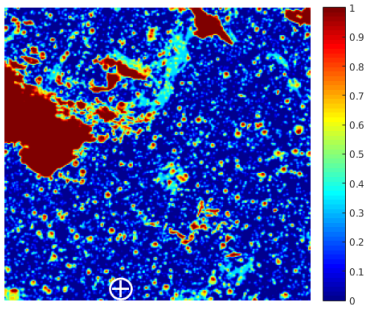
the edge detector based on the Laplacian of Gaussian filter: edges in general are sharply identified. In the particular case of craters with high Sun elevation , the boolean output of the edge detector does not recognizes the center of a small crater as hazardous in the case of high Sun elevation because of the low pixel intensity variation if no shadows are present. Indeed, it is able to identify as hazardous only



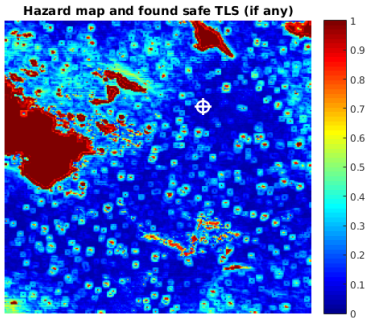
(a) True hazard map, test image 3



(b) Computed hazard map, test image 3



(c) True hazard map, test image 4



(d) Computed hazard map, test image 4

Figure 6.49: Comparison between ground truth and HD version E hazard map.

the edge of the crater.

Version F

Hazard map threshold: 0.30.
τ is 5.61 for a CPU time of 34.47 s. Thus version F is computationally heavier than the definitive HD system.

Hazard map

Hazard map RMSE of the HD architecture in Section 5.6 is depicted in Table 6.21 and a comparison with the definitive version is presented in Figure 6.50. Root Mean Square Error is much lower with

Test image	**Sun elevation [$^\circ$]**	**RMSE [-]**
1	80	0.02833
2	15	0.054577
3	80	0.034552
4	15	0.060832
5	80	0.038253
6	15	0.07037
7	80	0.028943
8	15	0.04721
Mean (Sun 15°)		**0.058247**
Mean (Sun 80°)		**0.032520**
Mean (all)		**0.045384**
Max (all)		**0.07037**
Min (all)		**0.02833**

Table 6.21: Hazard map RMSE, version F.

respect to the well performing version E. The maximum, registered in test image 7, is a mere 7%. Nevertheless, it ought be reminded that RMSE represents the data deviation with respect to the regression line. Hence, looking at Figures 6.51 and 6.52, data is very "packed" about the regression line, but the angular coefficient of this very last line is quite low, indicating a difficulty for HD version E to grasp image features at higher hazard indices, or in other terms, an underestimation of the high hazardous areas in the images.

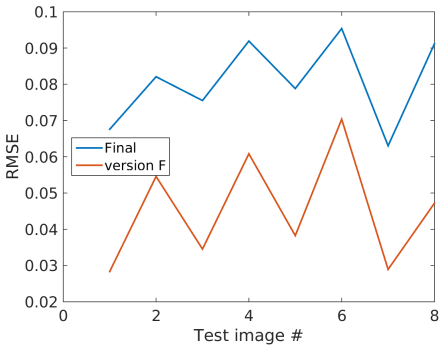
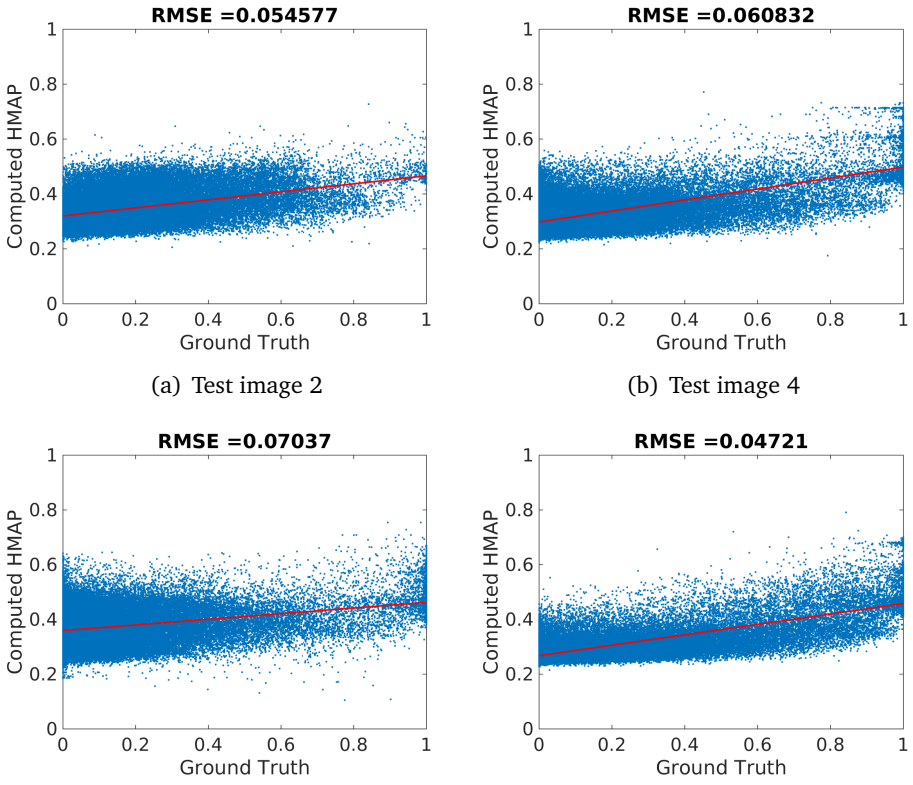


Figure 6.50: RMSE comparison between final HD architecture and version F on the test dataset.



(a) Test image 2

(b) Test image 4

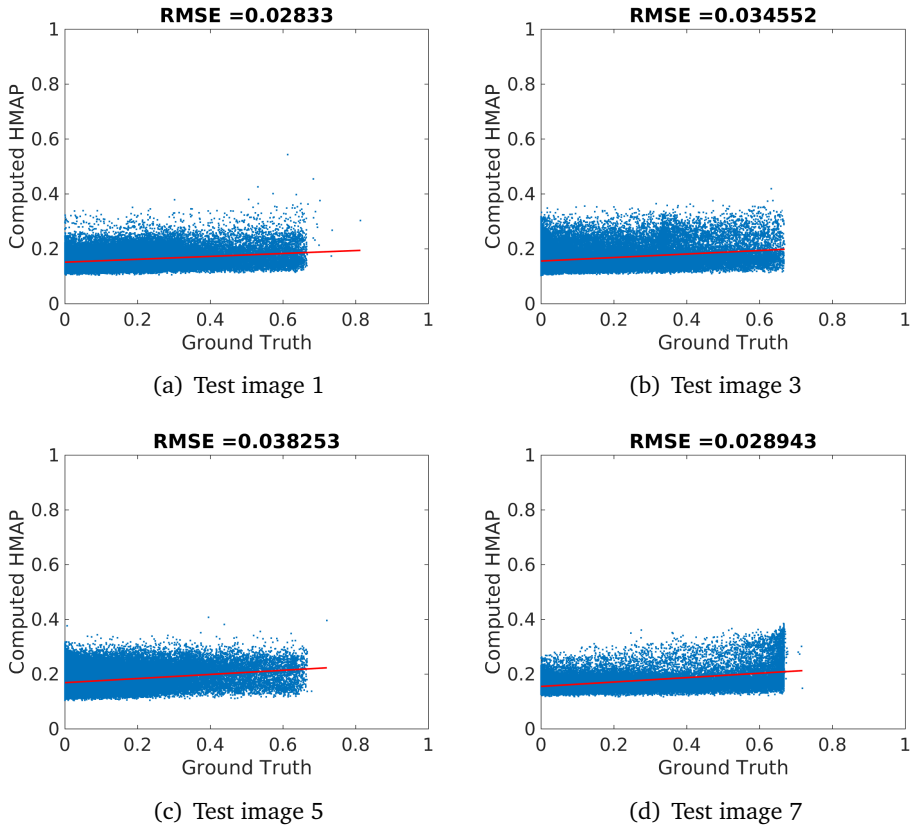
(c) Test image 6

(d) Test image 8

Figure 6.51: RMSE, version F: images with 15° Sun elevation in the test dataset.

Landing sites

Landing sites computation performances are depicted in Table 6.22, while in Figure 6.53 a comparison with the final HD system is pre-



(a) Test image 1

(b) Test image 3

(c) Test image 5

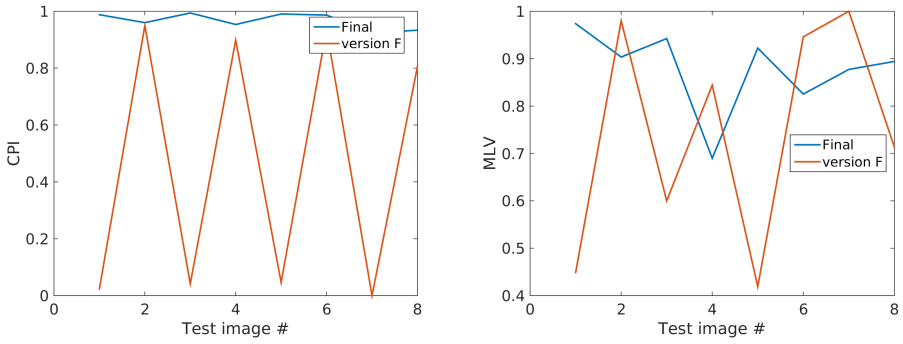
(d) Test image 7

Figure 6.52: RMSE, version F: images with 80° Sun elevation in the test dataset.

sented. The HD system did not found any true positive landing site for test image 7. Therefore all the performance computations involving such quantity are performed on the remaining test dataset images and they are signed with the star symbol(*). Dramatic failure occurs during the analysis of the 80°subset: CPI average remains low at 0.28, without any landing site found in test image 7. Performances on the low inclination subset are definitely better, with a CPI of 0.81.
MLV parameter oscillates between 0.42 and 0.98 for an average of 0.71, therefore not a particularly bad result, compared to the previous versions. In any case, due to the very poor performances registered for the CPI of the high Sun elevation angle test images, version F does not meet the system requirements.

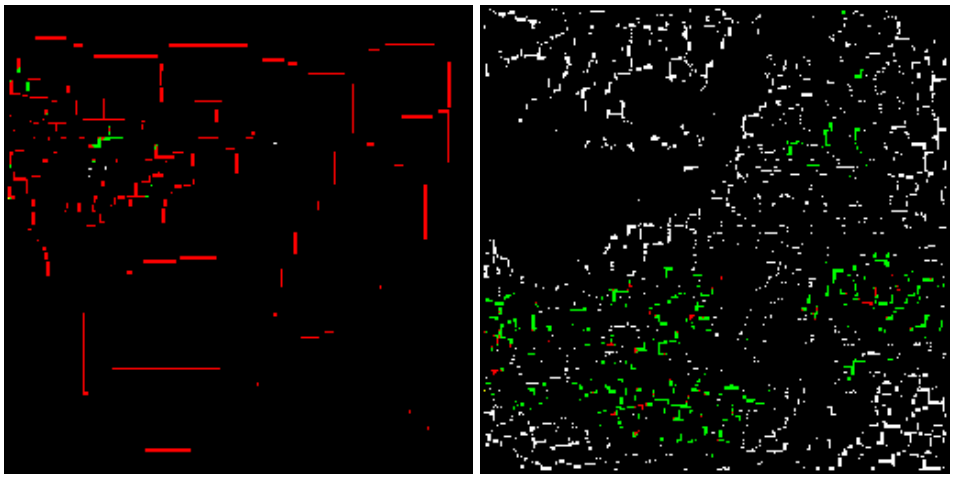
Test image	Sun elevation [°]	CPI [-]	MLV [-]
1	80	0.023836	0.448718
2	15	0.949580	0.979943
3	80	0.041695	0.600000
4	15	0.898386	0.844126
5	80	0.046346	0.418994
6	15	0.936170	0.946029
7	80	0*	-*
8	15	0.805596	0.713463
Mean (Sun 15°)		**0.897433**	**0.870890**
Mean (Sun 80°)		**0.027969**	**0.48924***
Mean (all)		**0.462701**	**0.70732***
Max (all)		**0.949580**	**0.979943**
Min (all)		**0**	**0.418994**

Table 6.22: Hazard detection system performances, version F.



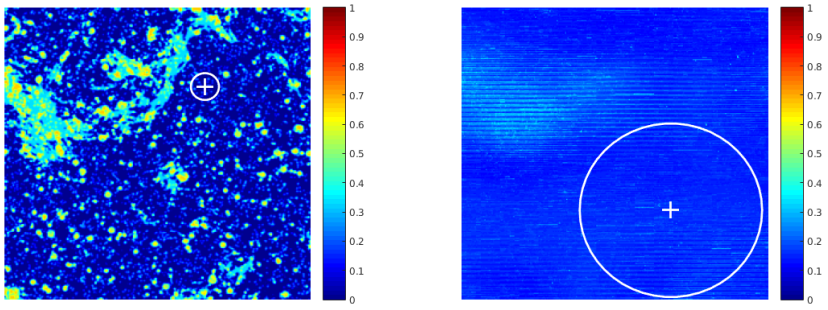
(a) CPI (b) MLV

Figure 6.53: Comparison between final HD version and version F in terms of CPI and MLV.

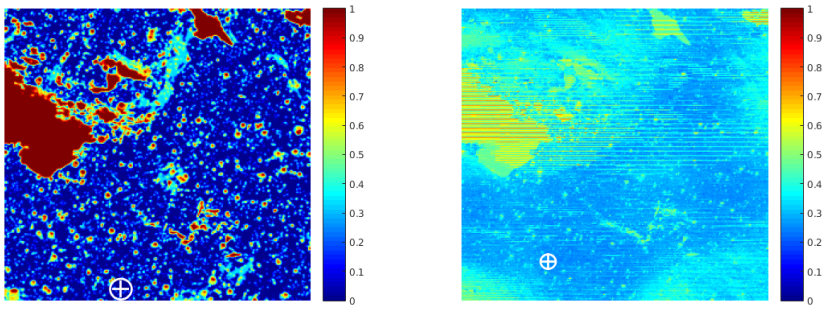


(a) Landing Sites, test image 3 (b) Landing Sites, test image 4

Figure 6.54: Resulting landing sites on test images 3 and 4. Green represents a True Positive, red a False Positive, white a False Negative.



(a) True hazard map, test image 3 (b) Computed hazard map, test image 3



(c) True hazard map, test image 4 (d) Computed hazard map, test image 4

Figure 6.55: Comparison between ground truth and HD version F hazard map.

Version G

Hazard map threshold: 0.30.
τ is 41.29 for a CPU time of 253.94 s. Thus version G is computationally heavier than the definitive HD system.

Hazard map

Hazard map RMSE of the HD architecture in Section 5.6 is depicted in Table 6.23 and a comparison with the definitive version is presented in Figure 6.56. RMSE numerical results quality looks averagely good:

Test image	**Sun elevation [°]**	**RMSE [-]**
1	80	0.12604
2	15	0.10403
3	80	0.14903
4	15	0.1325
5	80	0.03877
6	15	0.13615
7	80	0.10778
8	15	0.13439
Mean (Sun 15°)		**0.12677**
Mean (Sun 80°)		**0.1054**
Mean (all)		**0.11609**
Max (all)		**0.14903**
Mi (all)n		**0.03877**

Table 6.23: Hazard map RMSE, version G.

error remains 10% for the high inclination subset, 13% for the other. A strange behavior, with respect to the other 7 pictures is recorded in test image 5, where the regression line (Fig.6.58) resulted nearly horizontal with a very high y-axis interception point. This is a clear proof of the complete misunderstanding by the network of such test image: the hazard index in the computed hazard map is bounded between about 0.6 and 0.8, while the ground truth hazard indices sweep from 0 to 0.7.

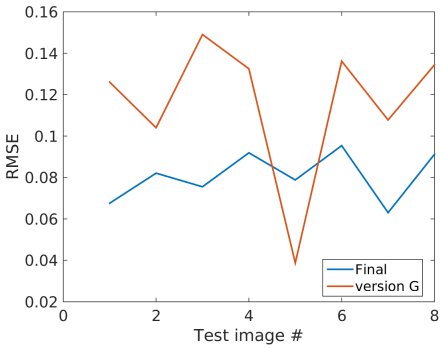
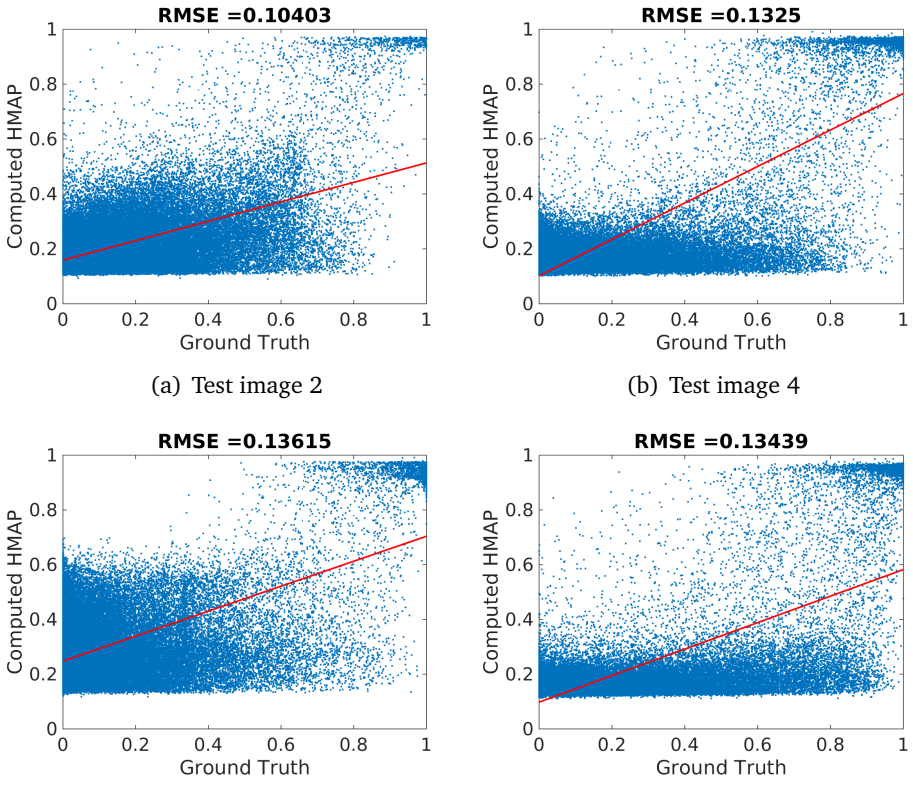


Figure 6.56: RMSE comparison between final HD architecture and version G on the test dataset.



(a) Test image 2

(b) Test image 4

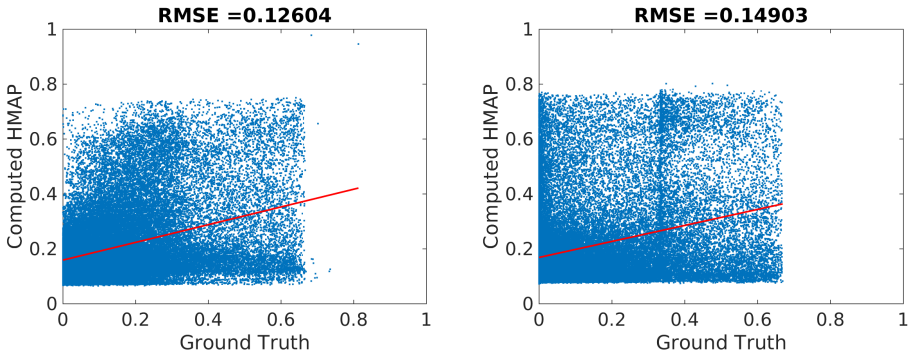
(c) Test image 6

(d) Test image 8

Figure 6.57: RMSE, version G: images with 15° Sun elevation in the test dataset.

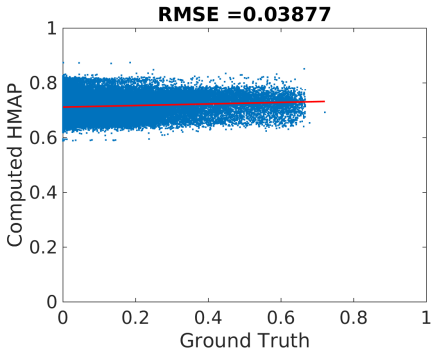
Landing sites

Landing sites computation performances are depicted in Table 6.24, while in Figure 6.59 a comparison with the final HD system is pre-

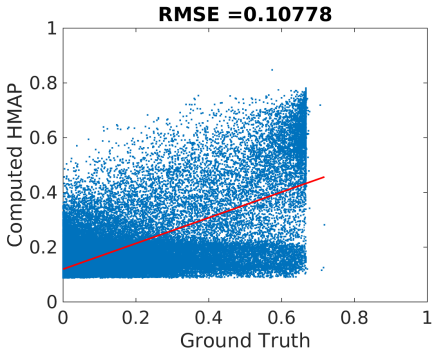


(a) Test image 1

(b) Test image 3



(c) Test image 5



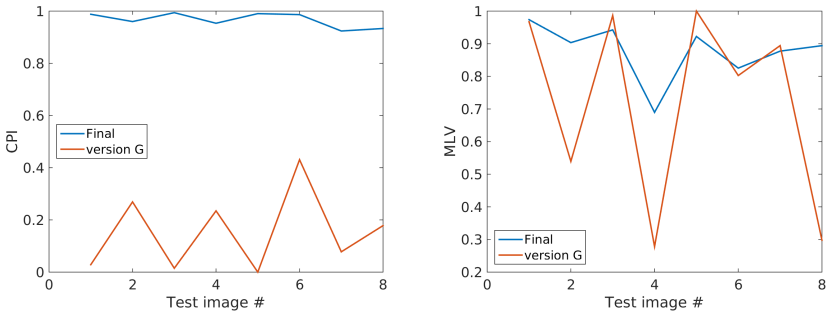
(d) Test image 7

Figure 6.58: RMSE, version G: images with 80° Sun elevation in the test dataset.

sented. The HD system did not found any landing site for test image 5. Therefore all the computations involving such quantity are performed on the remaining test dataset images and they are signed with the star symbol(*). Version G results one of the worst architecture variants developed: CPI total average, not considering test image 5, is a bare 0.18, making highly likely for the hazard detection system to choose a false positive as target landing site. In particular, performances on 80° test subset is tremendously bad: CPI amounts to 0.04. MLV records an average of 0.72, much increased by the ugly performance of the neural network on the test image 5, where no landing site at all was found.
Hazard detection version G architecture definitely does not meet the requirements.

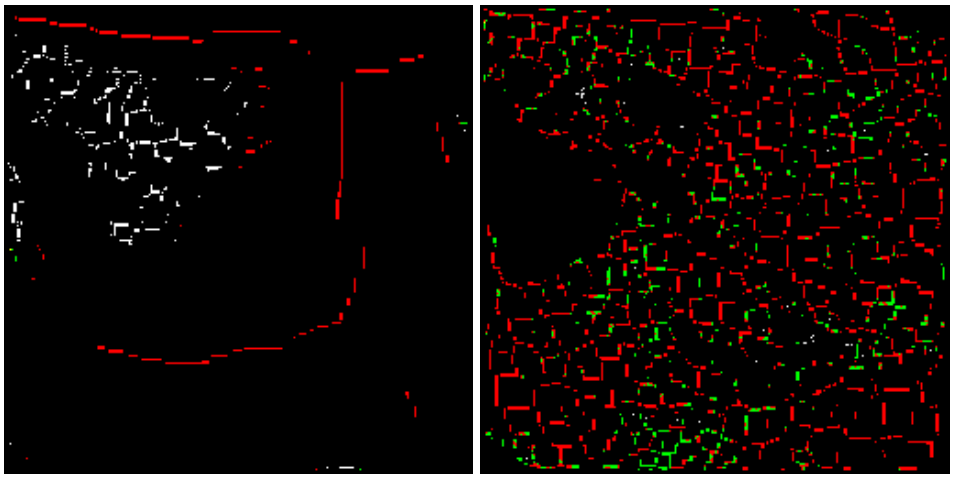
Test image	Sun elevation [°]	CPI [-]	MLV [-]
1	80	0.028421	0.967665
2	15	0.269051	0.539434
3	80	0.014771	0.986320
4	15	0.234721	0.278271
5	80	-*	1.000000
6	15	0.430739	0.802533
7	80	0.077778	0.894260
8	15	0.178231	0.297642
Mean (Sun 15°)		**0.278186**	**0.479470**
Mean (Sun 80°)		**0.040323***	**0.962061**
Mean (all)		**0.17624***	**0.720766**
Max (all)		**0.430739**	**1.000000**
Min (all)		**0.014771**	**0.278271**

Table 6.24: Hazard detection system performances, version G.



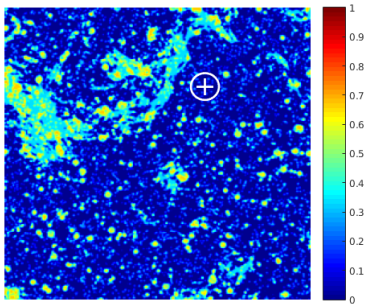
(a) CPI (b) MLV

Figure 6.59: Comparison between final HD version and version G in terms of CPI and MLV.

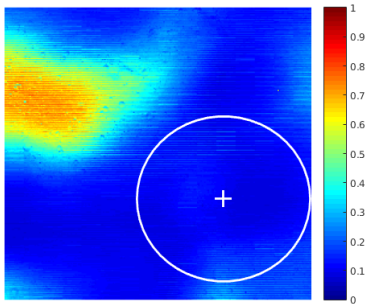


(a) Landing Sites, test image 3 (b) Landing Sites, test image 4

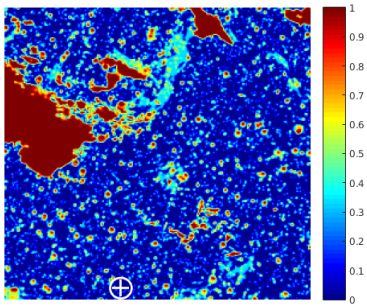
Figure 6.60: Resulting landing sites on test images 3 and 4. Green represents a True Positive, red a False Positive, white a False Negative.



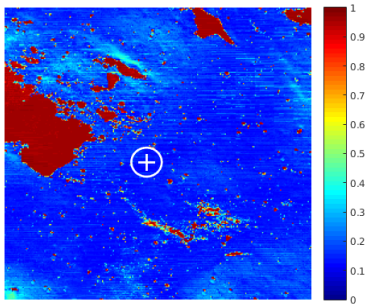
(a) True hazard map, test image 3



(b) Computed hazard map, test image 3



(c) True hazard map, test image 4



(d) Computed hazard map, test image 4

Figure 6.61: Comparison between ground truth and HD version G hazard map.

Version H

Hazard index threshold is set to 0.30.
τ is 11.28 for a CPU time of 69.40 s. Thus version H is computationally heavier than the definitive HD system.

Hazard map

Hazard map RMSE of the HD architecture in Section 5.6 is depicted in Table 6.25 and a comparison with the definitive version is presented in Figure 6.62. High Sun elevation images recorded a better RMSE

Table 6.25: Hazard map RMSE, version H.

Test image	**Sun elevation [°]**	**RMSE [-]**
1	80	0.060165
2	15	0.10087
3	80	0.074371
4	15	0.13123
5	80	0.12926
6	15	0.12793
7	80	0.066836
8	15	0.13299
Mean (Sun 15°)		**0.12326**
Mean (Sun 80°)		**0.082659**
Mean (all)		**0.10296**
Max (all)		**0.13299**
Min (all)		**0.060165**

value of 8.3% in average, with the other subset increasing the error to 12.4%. As it has been noted during the previous versions examinations, the RMSE only does not mean the goodness of the results. Indeed, it is possible to see from the RMSE plots in Figures 6.63 and 6.64 the angular coefficient of the regression line to be more similar to a 45° bisecting line in the case of 15° test subset, meaning that in such case the neural network can approximate with more precision all the various features present in the image, from the lowest to the highest hazard index.

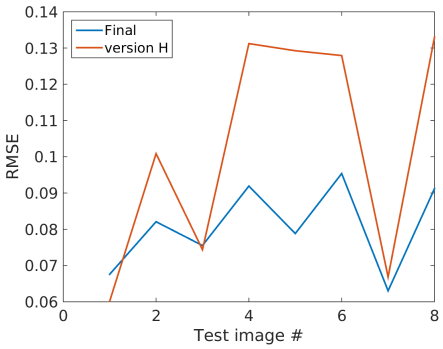
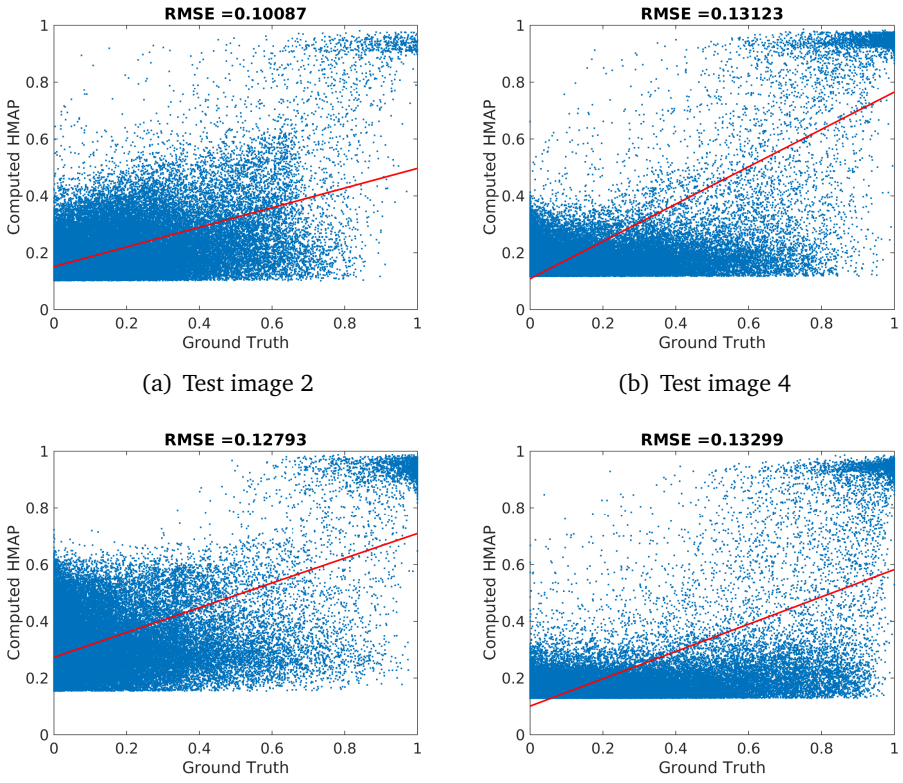


Figure 6.62: RMSE comparison between final HD architecture and version H on the test dataset.



(a) Test image 2

(b) Test image 4

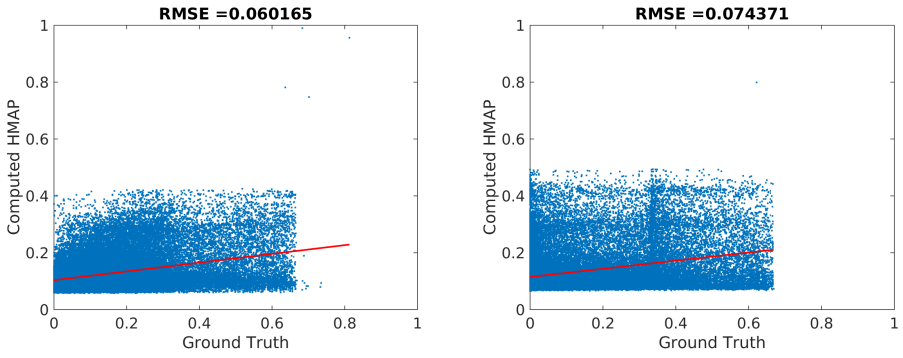
(c) Test image 6

(d) Test image 8

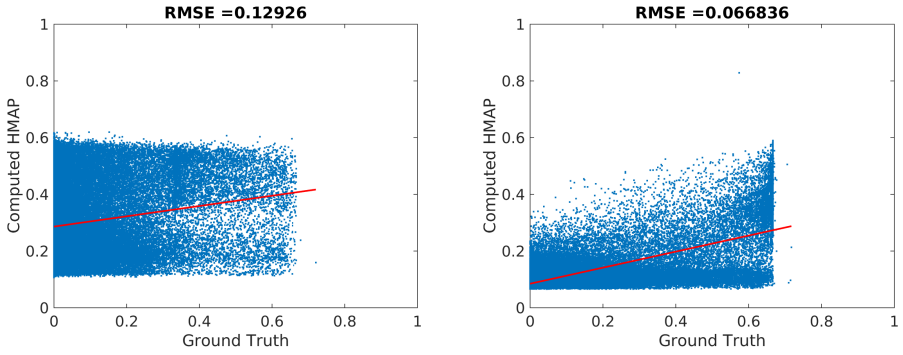
Figure 6.63: RMSE, version H: images with 15° Sun elevation in the test dataset.

Landing sites

Landing sites computation performances are depicted in Table 6.26, while in Figure 6.65 a comparison with the final HD system is pre-



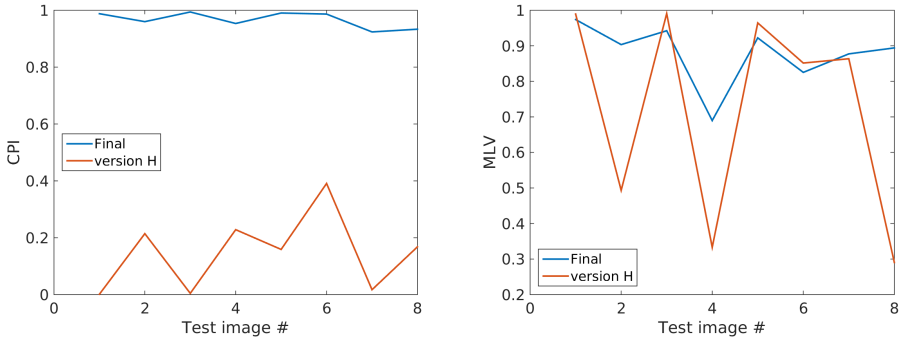
(a) Test image 1 (b) Test image 3



(c) Test image 5 (d) Test image 7

Figure 6.64: RMSE, version H: images with 80° Sun elevation in the test dataset.

sented. Landing site selection resembles HD version G, as it is pos-



(a) CPI (b) MLV

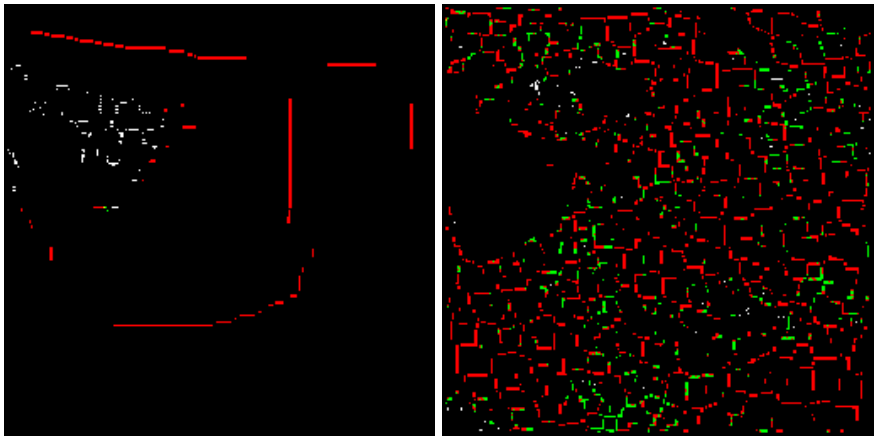
Figure 6.65: Comparison between final HD version and version H in terms of CPI and MLV.

sible to spot looking at Figures 6.61 and 6.67. Very bad general

Test image	Sun elevation [°]	CPI [-]	MLV [-]
1	80	0.001425	0.989362
2	15	0.214443	0.493287
3	80	0.004525	0.990712
4	15	0.228482	0.333154
5	80	0.158990	0.964452
6	15	0.390956	0.851547
7	80	0.017291	0.863636
8	15	0.168107	0.290434
Mean (Sun 15°)		**0.250497**	**0.492105**
Mean (Sun 80°)		**0.045558**	**0.952040**
Mean (all)		**0.148027**	**0.722073**
Max (all)		**0.390956**	**0.990712**
Min (all)		**0.001425**	**0.290434**

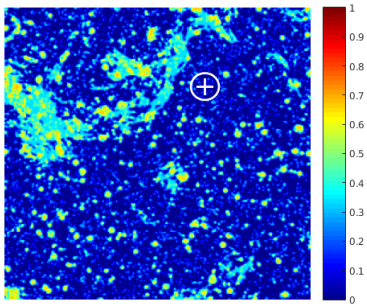
Table 6.26: Hazard detection system performances, version H.

average behavior, as the mean of CPI in the whole test dataset stops at 0.15, with the 80° subset results not reaching even for the 0.05. MLV oscillates from 0.99 to 0.29.
In any case, such low performances make the version H of the hazard detection system not able to meet the requirements.

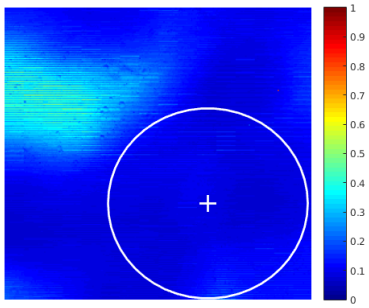


(a) Landing Sites, test image 3 (b) Landing Sites, test image 4

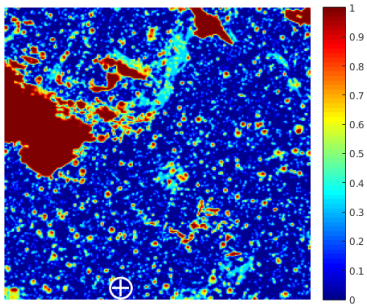
Figure 6.66: Resulting landing sites on test images 3 and 4. Green represents a True Positive, red a False Positive, white a False Negative.



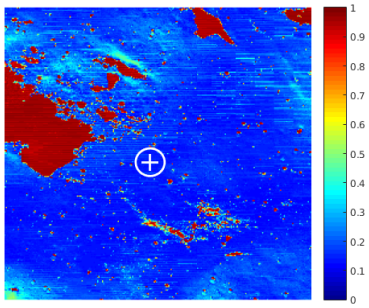
(a) True hazard map, test image 3



(b) Computed hazard map, test image 3



(c) True hazard map, test image 4



(d) Computed hazard map, test image 4

Figure 6.67: Comparison between ground truth and HD version H hazard map.

Version I

Hazard map threshold has been set to 0.22.
τ is 0.25 for a CPU time of 1.55 s. Thus version I is computationally lighter than the definitive HD system.

Hazard map

Hazard map RMSE of the HD architecture in Section 5.6 is depicted in Table 6.27 and a comparison with the definitive version is presented in Figure 6.68. Small RMSE (average: 0.04) values characterize the

Test image	**Sun elevation [$^\circ$]**	**RMSE [-]**
1	80	0.034152
2	15	0.11945
3	80	0.039452
4	15	0.16812
5	80	0.038324
6	15	0.1488
7	80	0.032687
8	15	0.13278
Mean (Sun 15°)		**0.14229**
Mean (Sun 80°)		**0.036154**
Mean (all)		**0.089221**
Max (all)		**0.16812**
Min (all)		**0.032687**

Table 6.27: Hazard map RMSE, version I.

high Sun elevation angle test subset. Nevertheless, from Figures 6.69 it can be seen that the regression line angular coefficient is very low. So far, every architecture with such characteristic has been found to bad perform. For what concerns the low Sun elevation angle test subset, RMSE grows up to a mean of 0.14 between the four images.

Landing sites

Landing sites computation performances are depicted in Table 6.28, while in Figure 6.71 a comparison with the final HD system is presented. Poor performances also for this HD version I: CPI for 15°

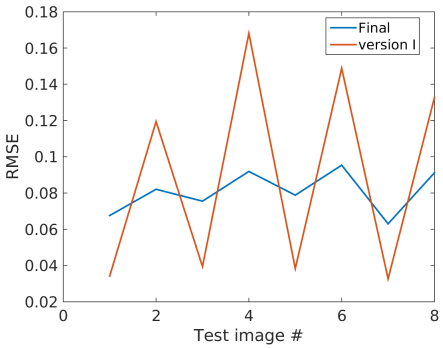
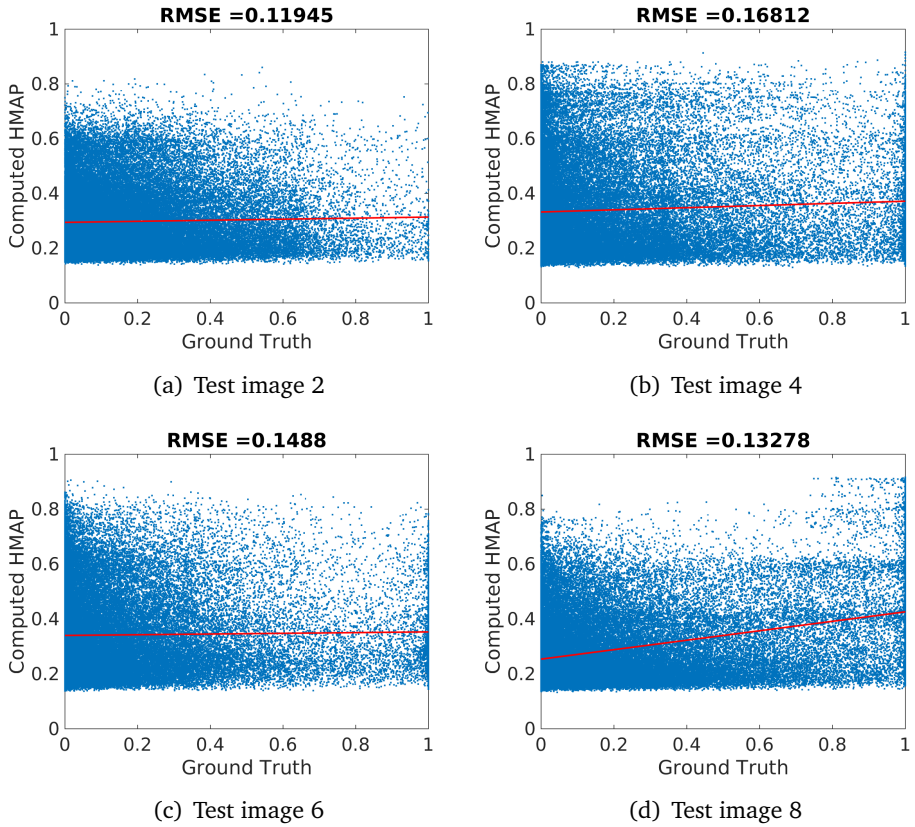


Figure 6.68: RMSE comparison between final HD architecture and version I on the test dataset.



(a) Test image 2

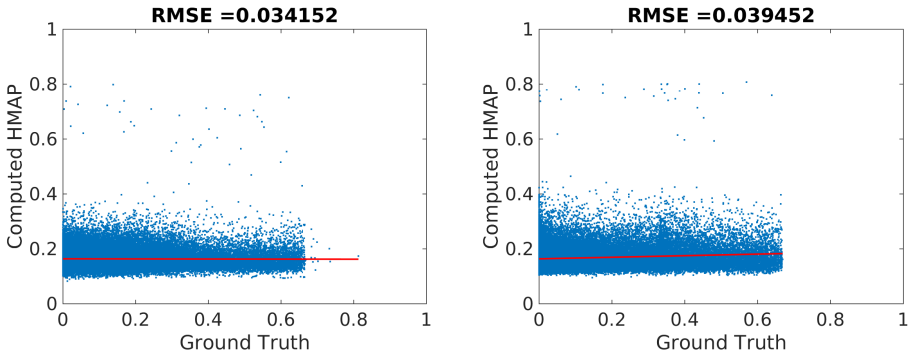
(b) Test image 4

(c) Test image 6

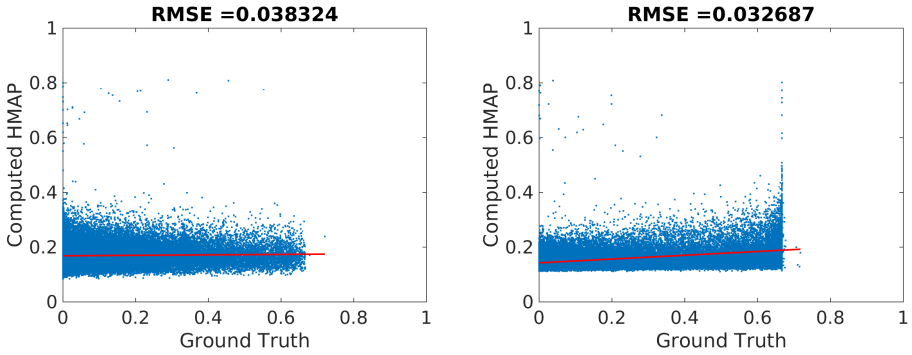
(d) Test image 8

Figure 6.69: RMSE, version I: images with 15° Sun elevation in the test dataset.

and 80° is respectively 0.50 and 0.24. Hence, this architecture has more probability to find a false positive than a true positive target landing site. MLV is significant, with an average of 0.81 among all

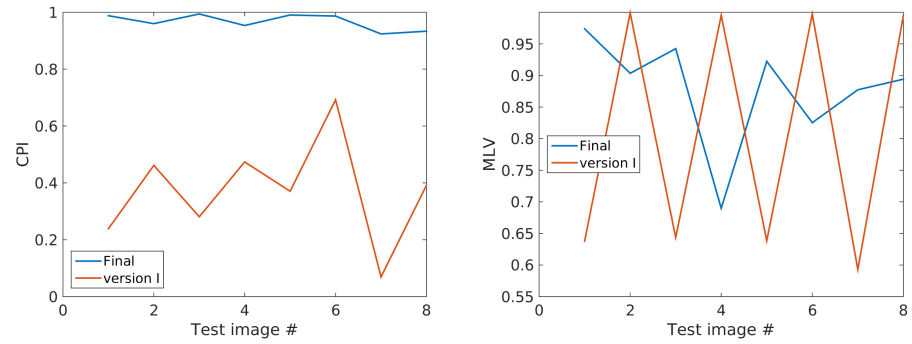


(a) Test image 1 (b) Test image 3



(c) Test image 5 (d) Test image 7

Figure 6.70: RMSE, version I: images with 80° Sun elevation in the test dataset.



(a) CPI (b) MLV

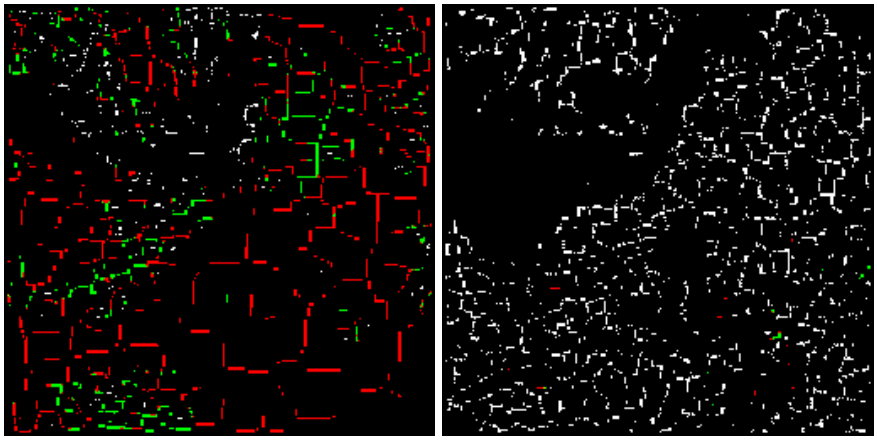
Figure 6.71: Comparison between final HD version and version I in terms of CPI and MLV.

the images of the test subset.
Due to the very low CPI values encountered, hazard detection system

Test image	Sun elevation [°]	CPI [-]	MLV [-]
1	80	0.238716	0.637596
2	15	0.461538	0.998921
3	80	0.280852	0.643610
4	15	0.473684	0.995115
5	80	0.370531	0.638401
6	15	0.692308	0.997156
7	80	0.069398	0.593074
8	15	0.391304	0.994620
Mean (Sun 15°)		**0.504709**	**0.996453**
Mean (Sun 80°)		**0.239874**	**0.628170**
Mean (all)		**0.372292**	**0.812312**
Max (all)		**0.692308**	**0.998921**
Min (all)		**0.069398**	**0.593074**

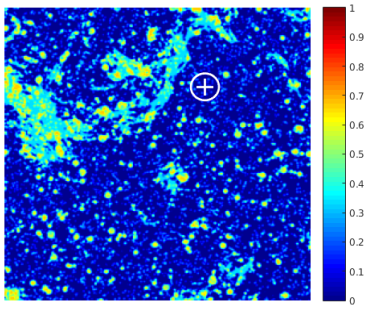
Table 6.28: Hazard detection system performances, version I.

architecture version I does not meet the requirements.

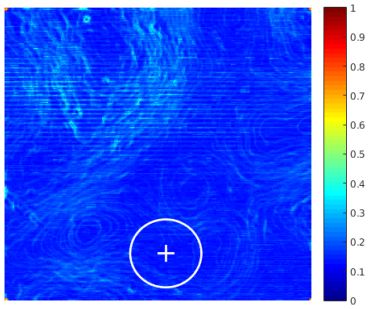


(a) Landing Sites, test image 3 (b) Landing Sites, test image 4

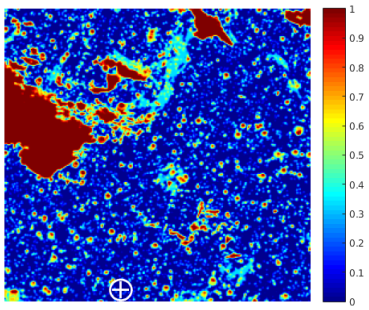
Figure 6.72: Resulting landing sites on test images 3 and 4. Green represents a True Positive, red a False Positive, white a False Negative.



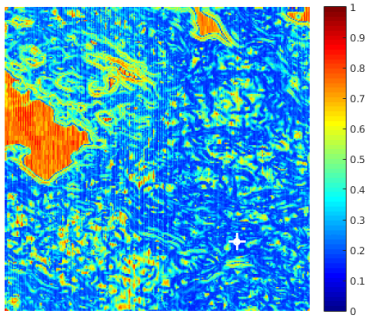
(a) True hazard map, test image 3



(b) Computed hazard map, test image 3



(c) True hazard map, test image 4



(d) Computed hazard map, test image 4

Figure 6.73: Comparison between ground truth and HD version I hazard map.

Version L

Hazard map threshold is set to 0.30.
τ is 0.91 for a CPU time of 5.6 s. Thus version L is slightly computationally lighter than the definitive HD system.

Hazard map

Hazard map RMSE of the HD architecture in Section 5.6 is depicted in Table 6.29 and a comparison with the definitive version is presented in Figure 6.74. Table 6.29 shows an average error of 20%, with a 23% registered for the low Sun elevation angle dataset. Looking at the data dispersion in Figure 6.75 and 6.76 it is possible to notice that the points tend to cluster about certain hazard values, probably because of the presence of the Kohonen networks in the architecture. In any case, regression lines have often a very high interception point with the ordinate axis plus a low inclination, symptom of a bad recognition of the surface hazardousness.

Test image	**Sun elevation [°]**	**RMSE [-]**
1	80	0.19869
2	15	0.26633
3	80	0.19381
4	15	0.24503
5	80	0.17656
6	15	0.25591
7	80	0.12923
8	15	0.15783
Mean (Sun 15°)		**0.23127**
Mean (Sun 80°)		**0.17457**
Mean (all)		**0.20292**
Max (all)		**0.26633**
Min (all)		**0.12923**

Table 6.29: Hazard map RMSE, version L.

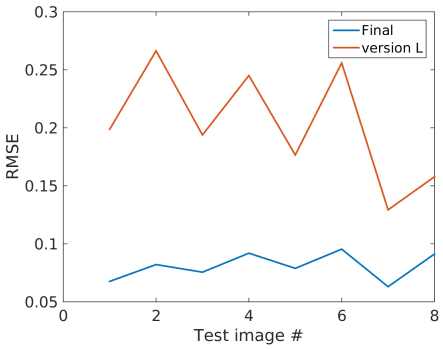
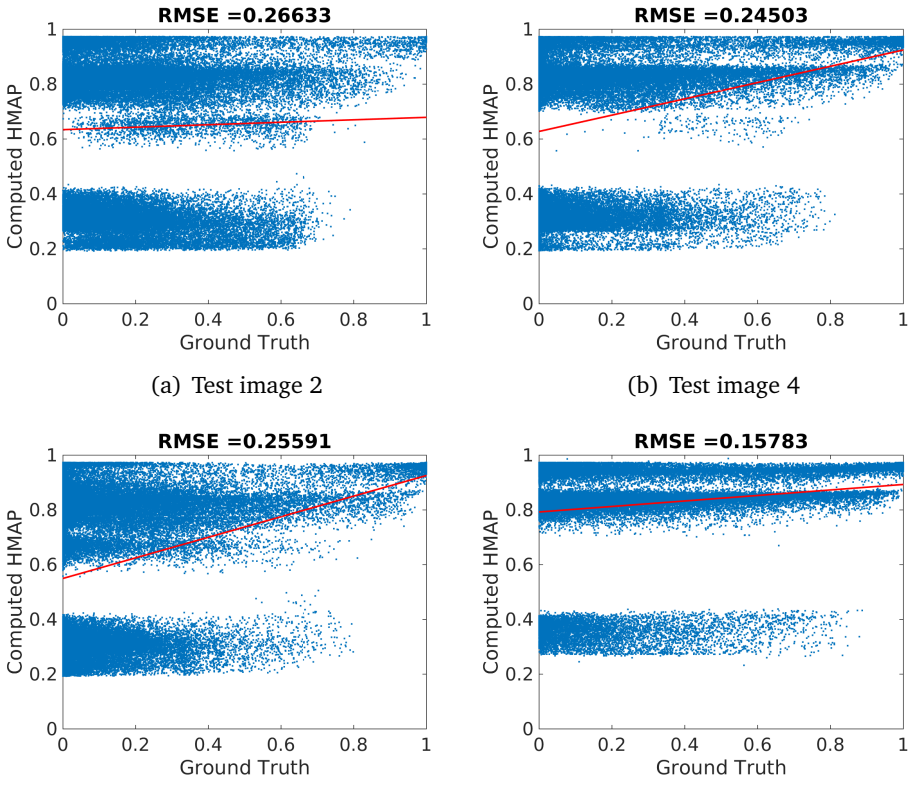


Figure 6.74: RMSE comparison between final HD architecture and version L on the test dataset.



(a) Test image 2

(b) Test image 4

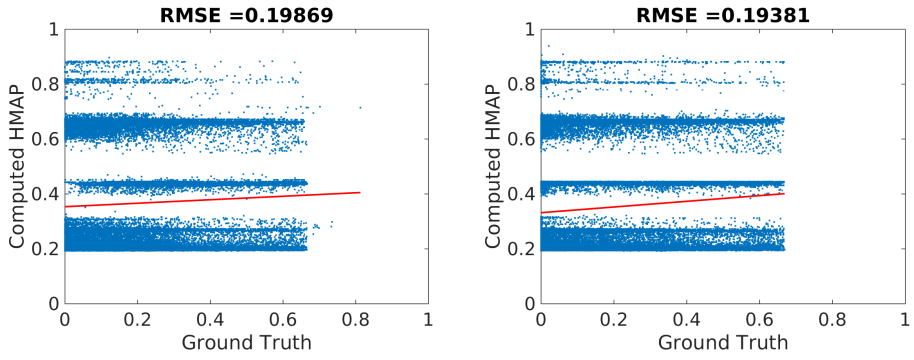
(c) Test image 6

(d) Test image 8

Figure 6.75: RMSE, version L: images with 15° Sun elevation in the test dataset.

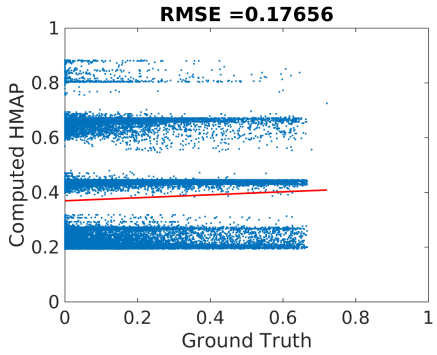
Landing sites

Landing sites computation performances are depicted in Table 6.30, while in Figure 6.77 a comparison with the final HD system is pre-

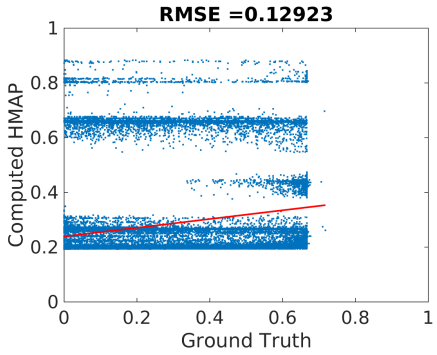


(a) Test image 1

(b) Test image 3



(c) Test image 5



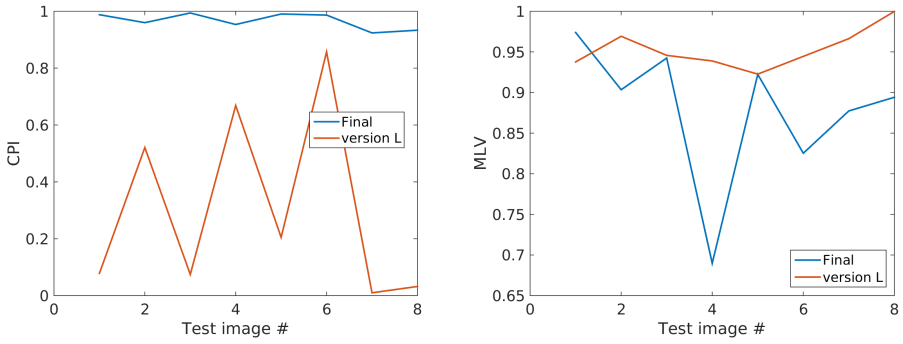
(d) Test image 7

Figure 6.76: RMSE, version L: images with 80° Sun elevation in the test dataset.

sented. CPI is low in any case apart from test image 6. The average for the 15°subset stops at 0.52, meaning roughly a landing site out of two is a false positive is extremely dangerous. Not to mention the terrible 0.09 for the high Sun inclination test subset in which is statistically almost certain to select a false positive as landing site. MLV are high, with an average of 0.95. Version L is clearly not suitable as an hazard detection system and it does not meets the requirements.

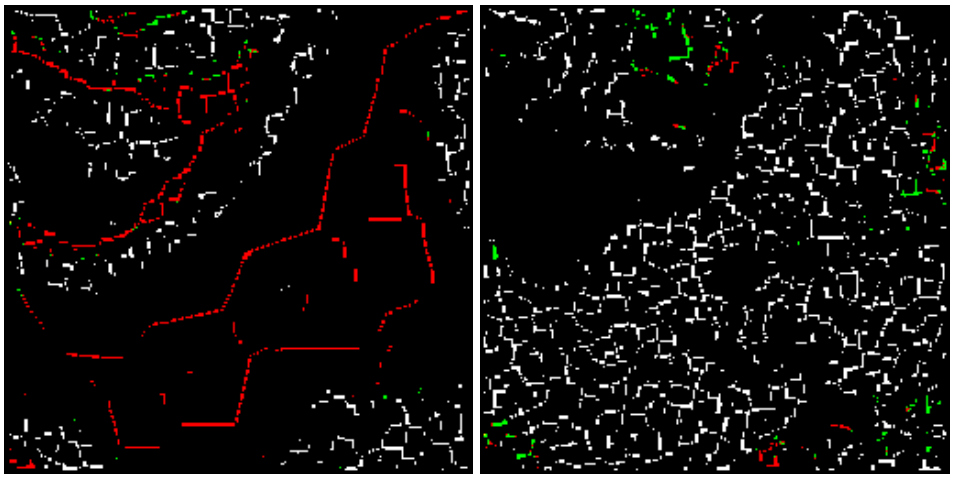
Test image	Sun elevation [°]	CPI [-]	MLV [-]
1	80	0.078358	0.937662
2	15	0.520833	0.969212
3	80	0.074504	0.945744
4	15	0.668593	0.938865
5	80	0.204940	0.922651
6	15	0.857143	0.944478
7	80	0.010000	0.966292
8	15	0.032258	0.999800
Mean (Sun 15°)		**0.519707**	**0.963089**
Mean (Sun 80°)		**0.091951**	**0.943087**
Mean (all)		**0.305829**	**0.953088**
Max (all)		**0.857143**	**0.999800**
Min (all)		**0.010000**	**0.922651**

Table 6.30: Hazard detection system performances, version L.



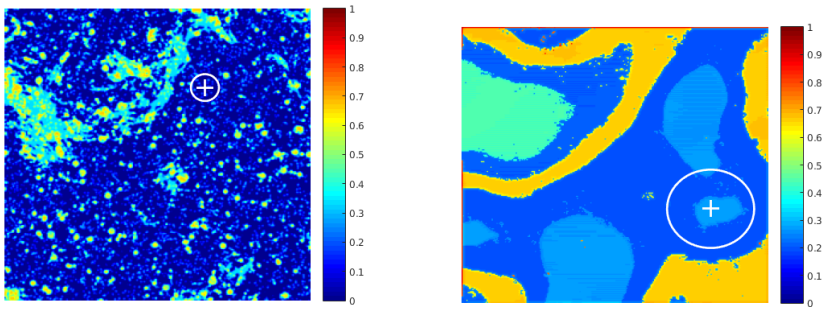
(a) CPI (b) MLV

Figure 6.77: Comparison between final HD version and version L in terms of CPI and MLV.

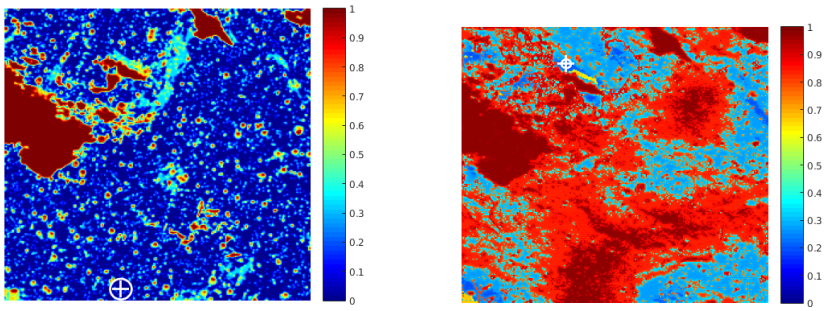


(a) Landing Sites, test image 3 (b) Landing Sites, test image 4

Figure 6.78: Resulting landing sites on test images 3 and 4. Green represents a True Positive, red a False Positive, white a False Negative.



(a) True hazard map, test image 3 (b) Computed hazard map, test image 3



(c) True hazard map, test image 4 (d) Computed hazard map, test image 4

Figure 6.79: Comparison between ground truth and HD version L hazard map.

Version M

Hazard map threshold is set to 0.25.
τ is 0.93 for a CPU time of 5.75 s. Thus version B is slightly computationally lighter than the definitive HD system. The three SOMs have dimension 10×10.

Hazard map

Hazard map RMSE of the HD architecture in Section 5.6 is depicted in Table 6.31 and a comparison with the definitive version is presented in Figure 6.80. Table 6.31 shows a high RMSE average of 17%,

Test image	**Sun elevation [$^\circ$]**	**RMSE [-]**
1	80	0.093156
2	15	0.23059
3	80	0.11137
4	15	0.25558
5	80	0.15831
6	15	0.23041
7	80	0.059275
8	15	0.23458
Mean (Sun 15°)		**0.23779**
Mean (Sun 80°)		**0.10553**
Mean (all)		**0.17166**
Max (all)		**0.25558**
Min (all)		**0.059275**

Table 6.31: Hazard map RMSE, version M.

particularly increased by the 15° subset that records a 24% error. High elevation images seem to be better understood, as shown by the 10.5% error. Tendency to cluster about some hazard indices is present here too like in version L, sign that this behavior is determined by the presence of the Kohonen networks. The situation seems a bit better with respect to version L, not only because of the lower RMSE, but also because of the regression line inclination and interception with ordinate axis, apart from test image 2 (Fig. 6.81), where the regression line has a negative slope.

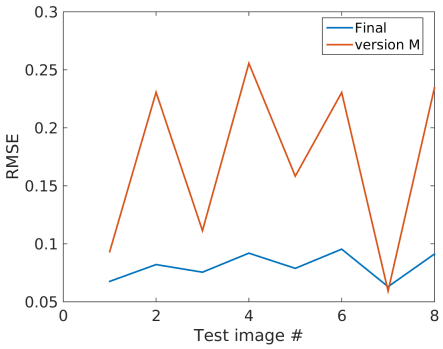
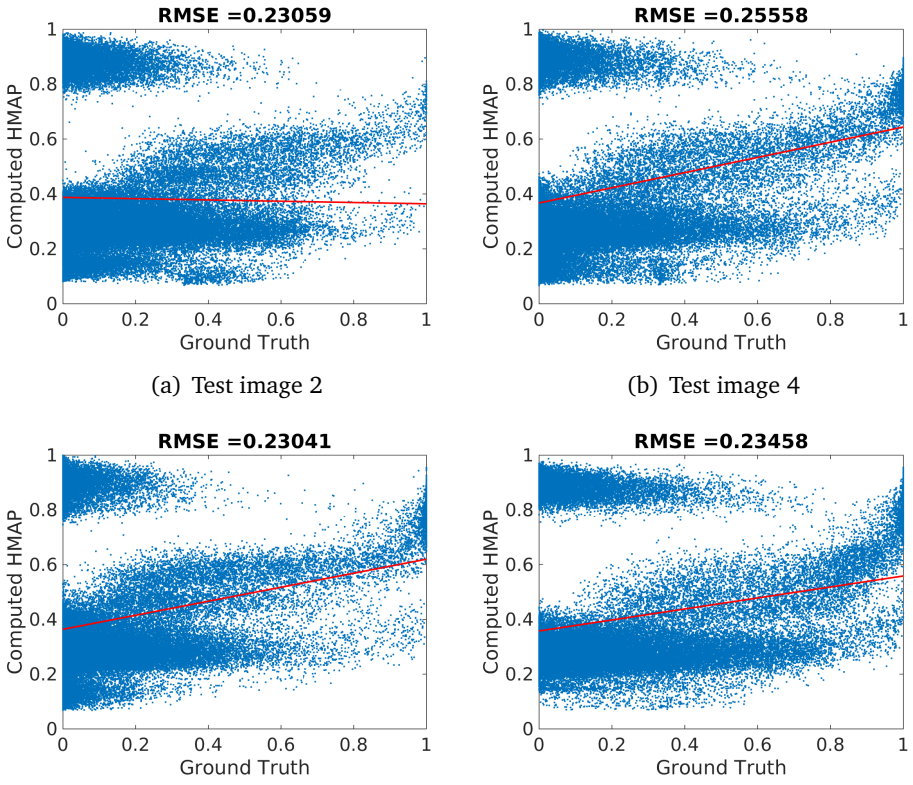


Figure 6.80: RMSE comparison between final HD architecture and version M on the test dataset.



(a) Test image 2

(b) Test image 4

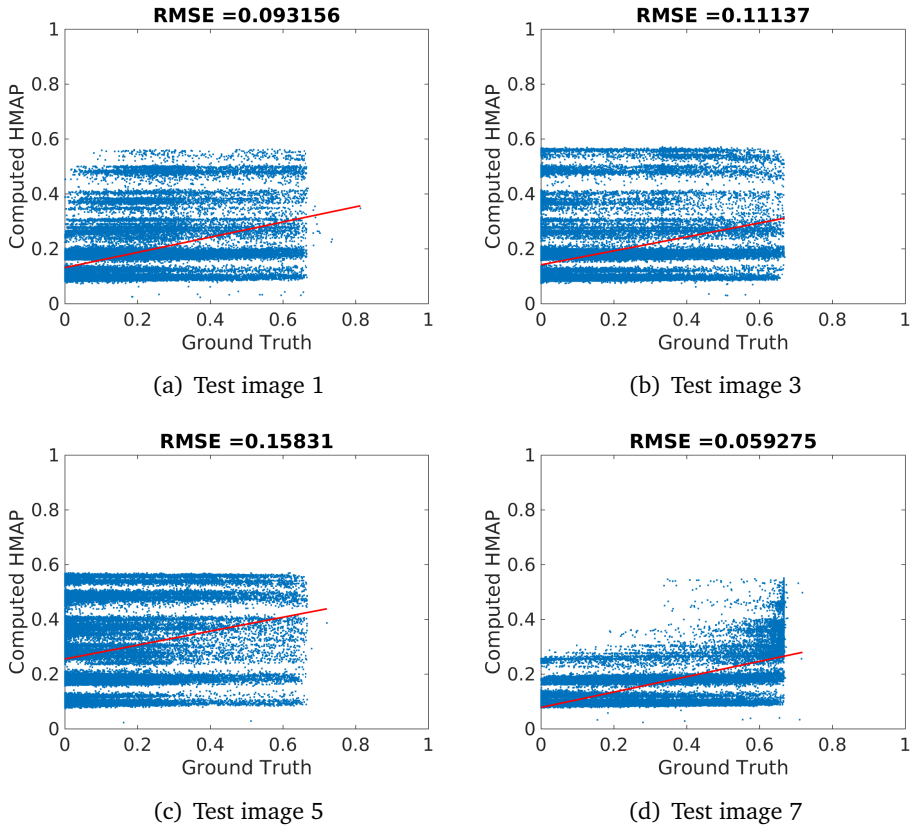
(c) Test image 6

(d) Test image 8

Figure 6.81: RMSE, version M: images with 15° Sun elevation in the test dataset.

Landing sites

Landing sites computation performances are depicted in Table 6.32, while in Figure 6.83 a comparison with the final HD system is pre-



(a) Test image 1

(b) Test image 3

(c) Test image 5

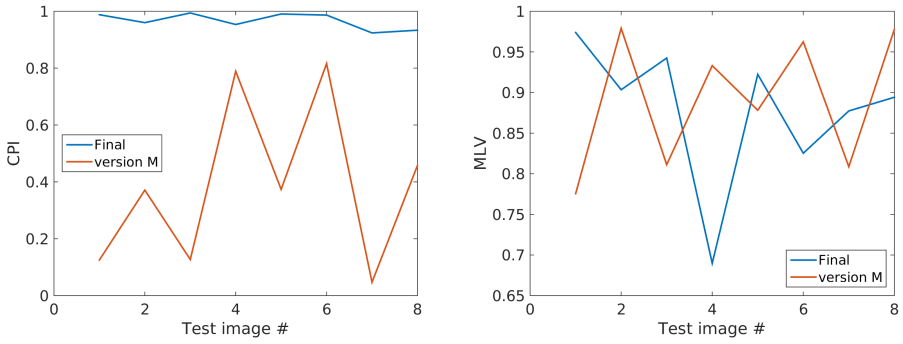
(d) Test image 7

Figure 6.82: RMSE, version M: images with 80° Sun elevation in the test dataset.

sented. A slight improvement with respect to version L, but totally unsafe in any case: for 80° subset the CPI is as low as 0.17. Situation is better for the other subset, where the HD version M reaches for 0.61, a value in any case too dangerous. MLV average amounts to 0.89. Because of the low CPI registered, version M does not meet the requirements.

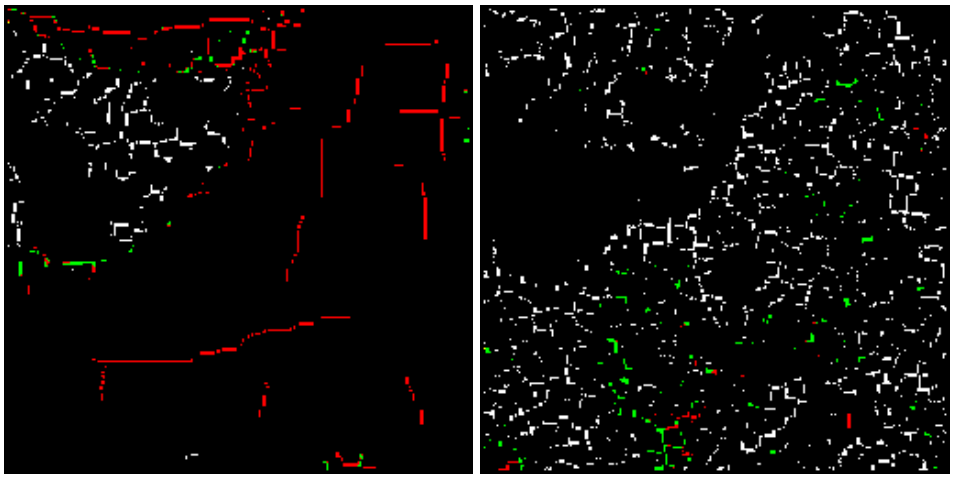
Test image	Sun elevation [°]	CPI [-]	MLV [-]
1	80	0.125366	0.775446
2	15	0.371069	0.979063
3	80	0.127104	0.811250
4	15	0.789047	0.933104
5	80	0.373568	0.878161
6	15	0.816054	0.962438
7	80	0.047210	0.808696
8	15	0.457490	0.977655
Mean (Sun 15°)		**0.608415**	**0.963065**
Mean (Sun 80°)		**0.168312**	**0.818388**
Mean (all)		**0.388364**	**0.890727**
Max (all)		**0.816054**	**0.979063**
Min (all)		**0.047210**	**0.775446**

Table 6.32: Hazard detection system performances, version M.



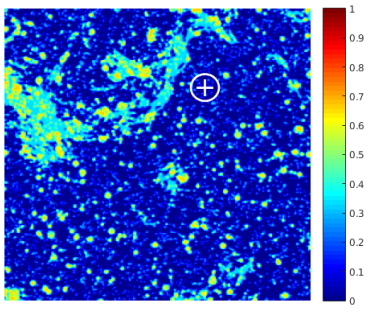
(a) CPI (b) MLV

Figure 6.83: Comparison between final HD version and version M in terms of CPI and MLV.

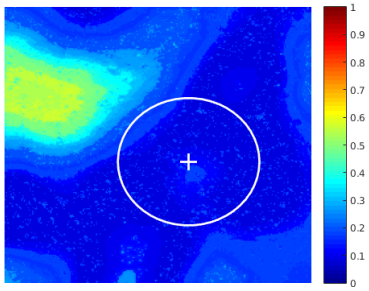


(a) Landing Sites, test image 3 (b) Landing Sites, test image 4

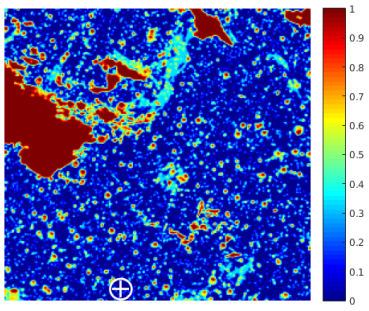
Figure 6.84: Resulting landing sites on test images 3 and 4. Green represents a True Positive, red a False Positive, white a False Negative.



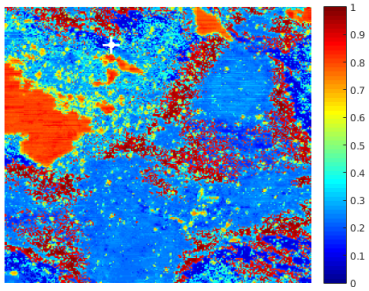
(a) True hazard map, test image 3



(b) Computed hazard map, test image 3



(c) True hazard map, test image 4



(d) Computed hazard map, test image 4

Figure 6.85: Comparison between ground truth and HD version M hazard map.

6.4.1 Comparison between final and discarded versions

Two bar graphs are presented to easily visualize the trend of the two most important parameters of merit for the various versions: CPI and τ. In particular the average among all the test set images is shown in Figure 6.86, and it is possible to notice that only the definitive hazard detection system is able to assess a CPI over 0.9, as requested by the system requirements. (Section 1.5)

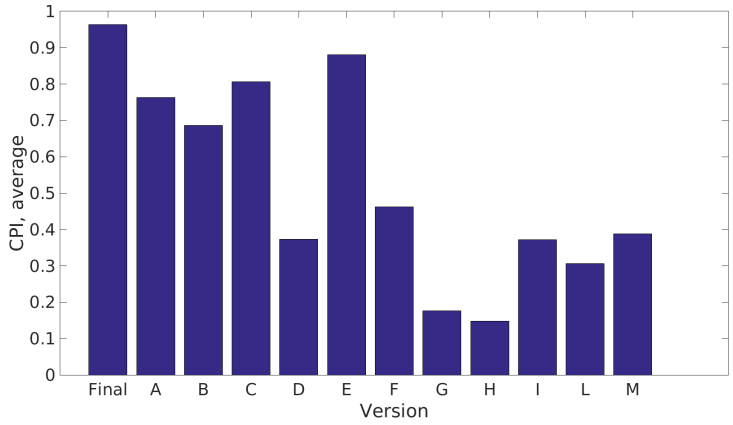


Figure 6.86: Trend of average CPI in the test set images

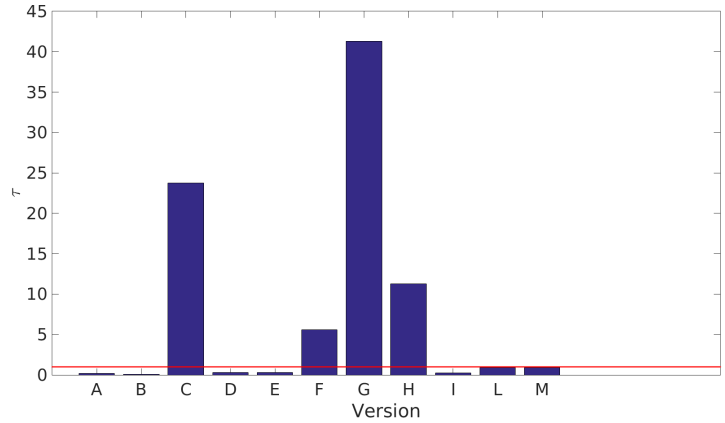


Figure 6.87: τ computational performance trend among the various architectures. Red horizontal line marks the reference $\tau = 1$ of the final HD version.

Looking at the graph in Figure 6.87, version C and E are the best among the discarded in terms of landing site candidates performances. Unfortunately, version C results too heavy computationally speaking with a τ of 23.77. Looking instead at the MLV trend in Figure 6.88, it is possible to notice that the final version does not score the lowest MLV among the HD versions. Nevertheless, it is worth to remind the lower importance of the MLV with respect to the CPI as performance indicator.

Focusing on version E, a sensitivity analysis of the landing site per-

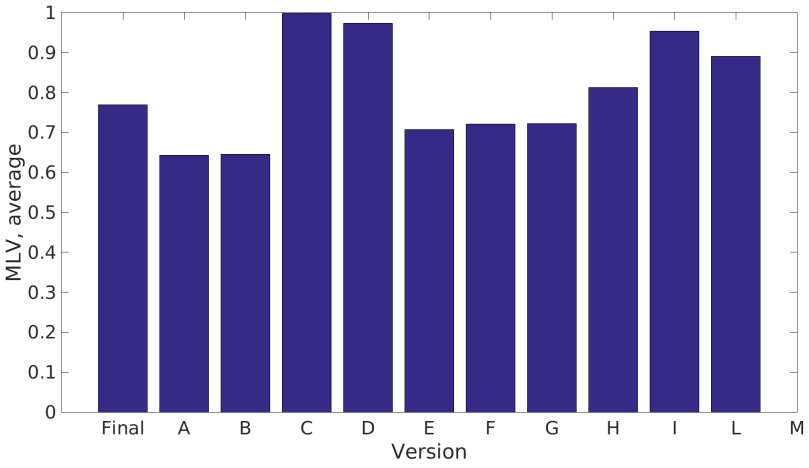
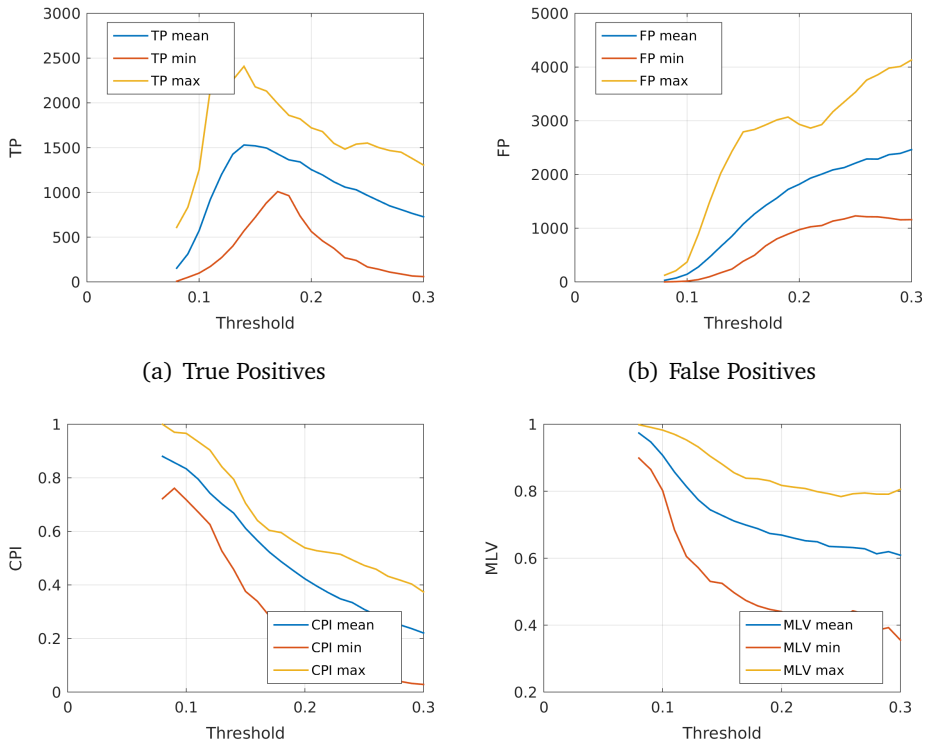


Figure 6.88: Trend of average MLV in the test set images

formances with respect to the threshold have been done to be sure that the definitive version of the hazard detection system is the best performing. Such analysis is presented in Figures 6.89: being the average CPI a decreasing monotonic function, maximum is reached at the minimum threshold of 0.09. It is possible to conclude without further doubts that the definitive version of the HD system is the best performing in terms of landing sites performances.



(a) True Positives

(b) False Positives

(c) CPI

(d) MLV

Figure 6.89: Trend of various HD system performances changing the threshold of the computed hazard map, discarded version E.

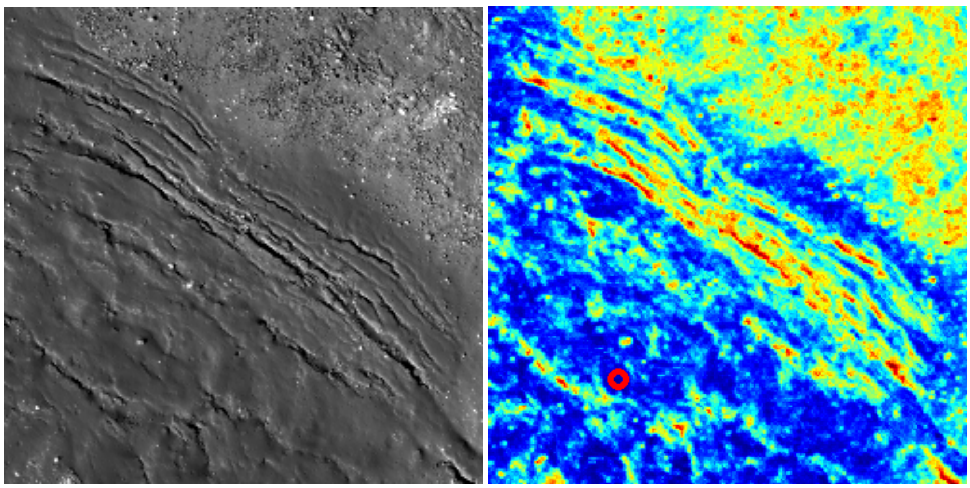
6.5 Test on real images

The very same neural network and algorithm has been also tested on real lunar images and photos taken by Rosetta mission of the 67P/Churyumov–Gerasimenko comet. Being unknown the Sun inclination angle, it has been briefly hypothesized looking at the photos, and this information lack is particularly critical for the performances of the HD system. Moreover, in these images there is no ground truth to quantitatively test the hazard maps with. Thus, results are to be intended just as an example of the generalization capabilities of the neural network and must be not intended as a valuable result of the proposed architecture. Anyway, the choice of the photos was dictated by the presence of relevant morphological features, that could have challenged the system.

Moon

In Figure 6.90, taken by LROC Narrow Angle Camera, depicting part of the Larmor Q crater floor, it is possible to spot some fractures on the surface in the lower left hand side half, while the rest of the image is characterized by diffuse roughness due to craters. In its relative hazard map (Fig. 5), the neural network seems to have qualitatively understood the terrain features, assigning a distributed high hazard value to the rough region at top right hand side and about maximum value precisely where fractures are located.

In Figure 6.91 the surface is characterized by a diffuse roughness. The small scale craters are recognized correctly as dangerous, such as the deep shadows. It must be pointed out that the quality of this image is quite low, and the shadow is not completely black in the original .png format gray intensity matrix. Indeed, in the two darkest spots in the bottom and in the left hand side of the image, such as the abrupt altitude change that cuts the image in anti-diagonal position, the highest hazard index is not assigned by the neural network, but it is in any case given a hazard index high enough for them not to be considered as possible landing sites.
Figure 6.92 represents a rough depression surrounded by what looks like safer planar areas. Although once again the network understand in general small scale hazards and rough regions, the evaluation of



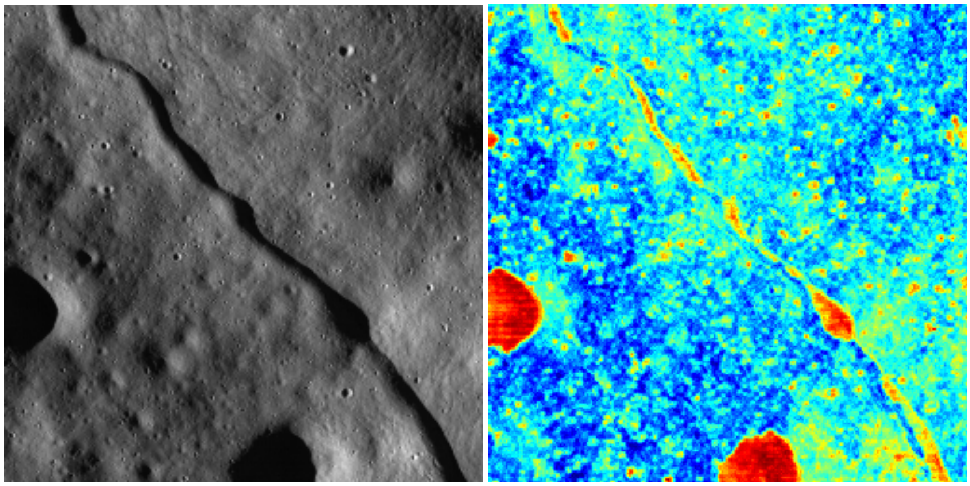
(a) Lunar image (b) Hazard map

Figure 6.90: Lunar real surface image, Larmor Q crater floor, NAC frame M151726155R, courtesy of NASA/GSFC/ASU.

the slope change in the central left hand side of the picture is more difficult to judge, where the neural network depicts as safe the darkest part, which looks like a small valley with a crater in the center and therefore could be an hazardous area. The smooth region in the bottom right hand side seems to be correctly identified as the safest area in the image.

Figure 6.93 looks in general well understood, apart from the upper left corner, where an apparent sloppy area is classified as safe. The same mistake happens in the bottom left corner in a small area on the left of the sharp edge marked correctly in red. The two main smooth planes in the upper half of the image are correctly understood: safe excluding the small craters and the edges present.

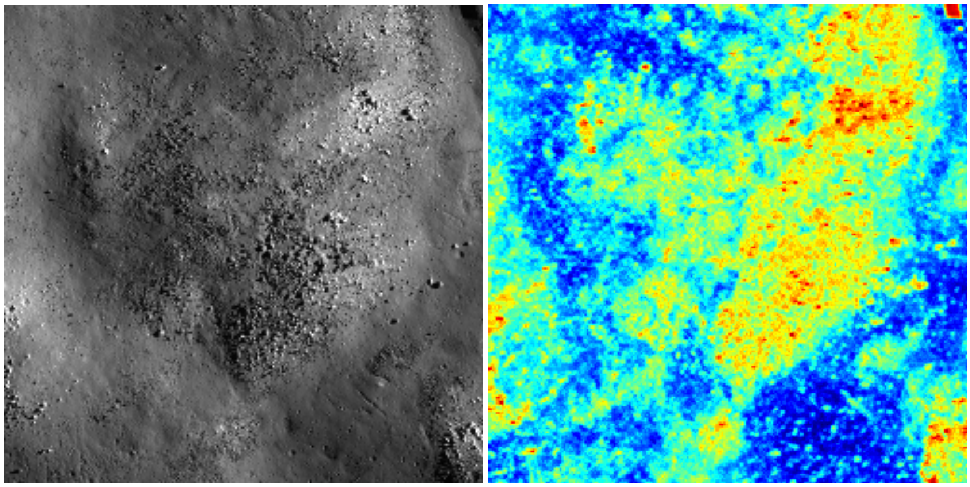
In Figure 6.94 to the deep shadow is not assigned a completely hazardous index, and this is probably due to the fact that the .png file was not composed of zeros in that area, but contained small non-null intensities that have fooled the network. The smoothest plane region nearby the image center is well considered safe, while rough planar areas have –again correctly– an intermediate hazard index. Abrupt altitude changes are recognized, such as the one in the central portion of the frame. In Figure 6.95 crater *Giordano Bruno* on the Moon is depicted in an oblique view. It is worth to point out that the neural network have been trained with images shot by a vertical attitude,



(a) Lunar image (b) Hazard map

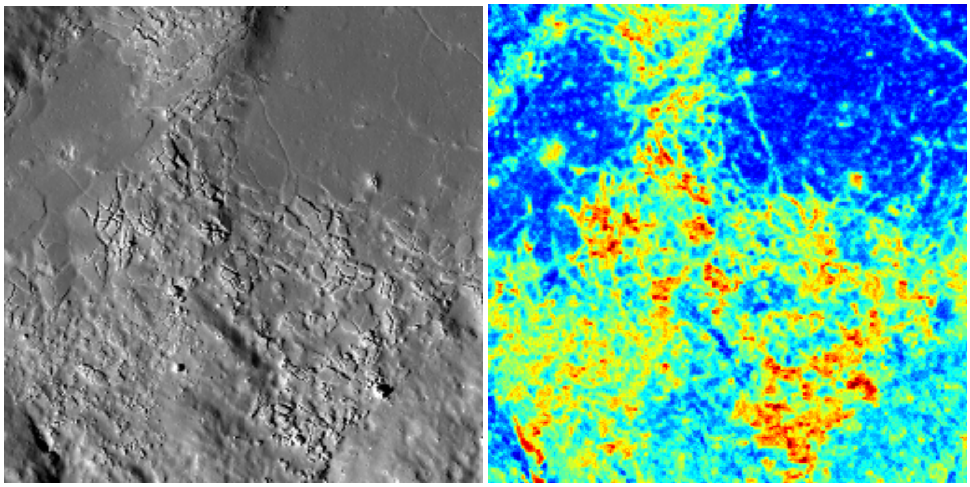
Figure 6.91: Real lunar test image

therefore this test case falls out of the operational range of the hazard detection system. In any case, for the sake of curiosity and to test the limit of the neural network comprehension capabilities, Figure 6.95 has been fed into the HD system. The corresponding hazard map – always Figure 6.95 – shows that the neural network is able to grasp the main features present: outer space in black is perfectly recognized, crater sides and edges have an high hazard index. Also slopes in foreground are marked in general as dangerous, though not as hazardous as it should be, especially small areas where the neural network thinks to be a landing suitable spot. The identified target site (small red circle), seems well recognized as a safe area.



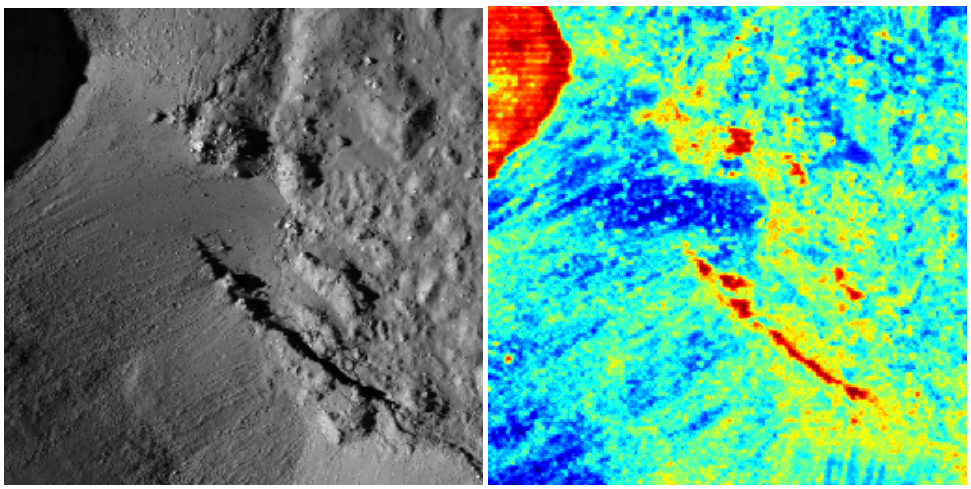
(a) Lunar image (b) Hazard map

Figure 6.92: Real lunar test image



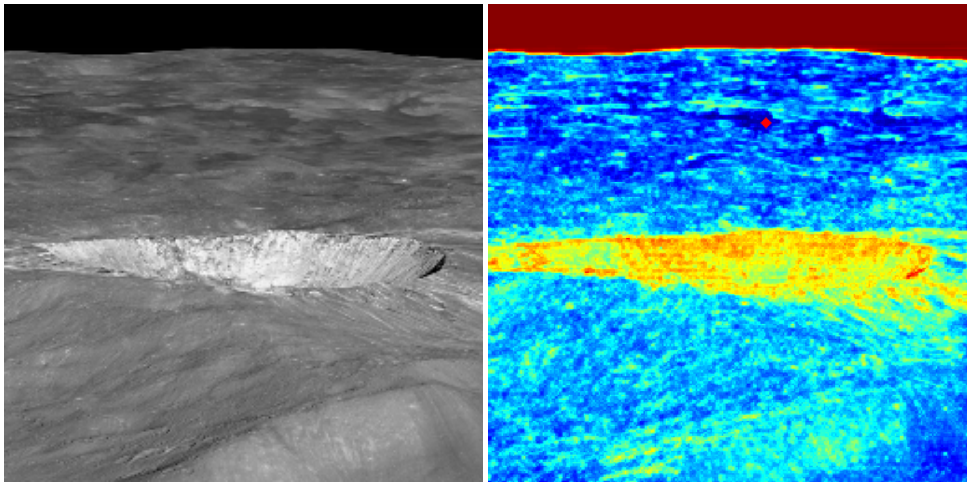
(a) Lunar image (b) Hazard map

Figure 6.93: Real lunar test image



(a) Lunar image (b) Hazard map

Figure 6.94: Real lunar test image



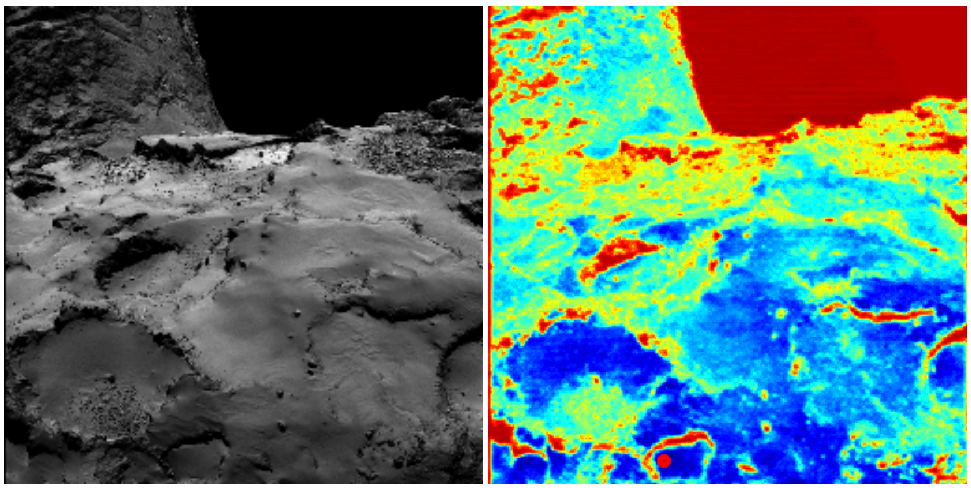
(a) Lunar image (b) Hazard map

Figure 6.95: Giordano Bruno crater, oblique view.

67P/Churyumov-Gerasimenko

The great interest of both the scientific community and private companies in small celestial bodies pushed to test the same hazard system used for lunar images on 67P/Churyumov-Gerasimenko. Not many suitable images are available for the purpose, and even less are equipped with data the neural network should need to be as much efficient as it can. The hazard detection system have been tested on 67P images to assess the robustness of the system in an ambient that is completely different from the one which it has been designed for, that is the Moon.

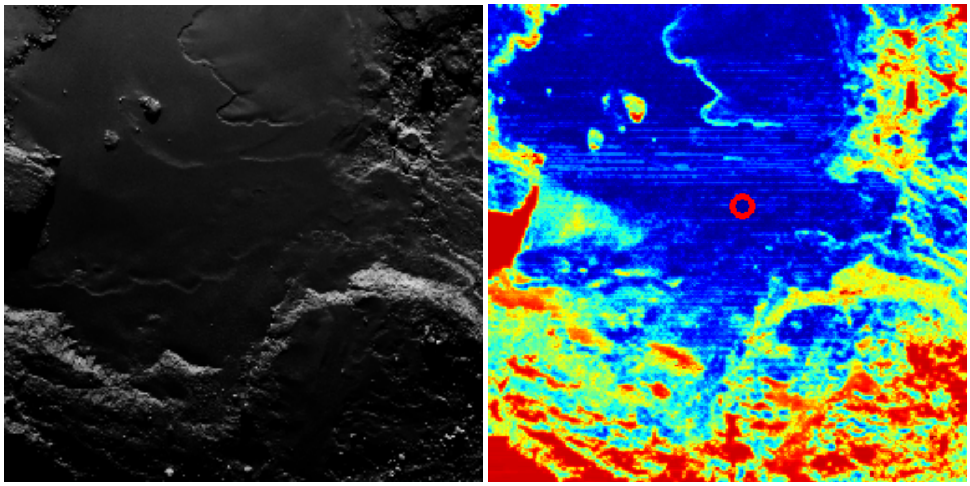
A first test is performed on a frame taken by Rosetta OSIRIS camera



(a) 67P image (b) Hazard map

Figure 6.96: Comet 67P Churyumov-Gerasimenko, Philae backup landing site C (Photo: ESA)

depicting the Philae backup landing site C. The image is quite different to the ones the neural network has been trained with, especially because of the OSIRIS camera optical characteristics, that are opposite to the pinhole simulated with POV-Ray. So said, once again the network understand the main morphological features: foreground planar areas are classified safe, their rough edges unsafe. Deep space in the top of the picture is correctly assumed unsafe, such as an intermediate dangerous classification is given to rough regions. Landing site found in the bottom region looks like a legitimate choice. Another test has been done on the Imhotep region of 67P comet and it is pre-



(a) 67P image (b) Hazard map

Figure 6.97: Comet 67P Churyumov-Gerasimenko, Imhotep region (Photo: ESA)

sented in Figure 6.97. Such an area is composed by many well distinct features: a planar plateau with sharp boulders and rifts, developing from the center to the top of the picture, surrounded by an irregular area full of craters and high sloped sides. In the relative computed hazard map, the system seems to have qualitatively understood hazard trends of the various areas: deep blue (safe) for the planar area apart from the irregularities, green and red (unsafe) for the most of the rest. Landing site selected seems adequately safe.

Chapter 7

Facility for autonomous planetary landing simulation

THE *design of a hardware-in-the-loop test environment dedicated to the analysis of optical planetary landing systems is presented. This facility will be hosted in PoliMi DAER premises and will be devoted to integration and validation of the various subsystems involved in the landing phase. Motivations and benefits to the hazard detection and avoidance system deriving from the facility are explicated. Two main design variants are taken into account and their pros and cons discussed.*

7.1 Motivations

An Hazard Detection and Avoidance system based on optical sensors requires some additional information regarding spacecraft attitude and trajectory, camera model and external environment. Space missions performed during the past years registered a great quantity of data, and a large portion is freely available to use. Unfortunately, they usually lack of most of the metadata fundamental to perform a proper simulation. Moreover, the largest part of planetary images available are taken with narrow angle cameras from great distances and thus they are not representative of the optical sensor that an hazard detection system exploits. Hence, it is difficult to exploit such data for testing and validation of hazard detection algorithms.
As it has been presented in Chapter 4, synthetic images have been

exploited to develop the hazard detection algorithms. Therefore, the system needs to be validated with experimental data to verify its behavior in an actual environment. Moreover, the various subsystems composing the HDA need to be tested to assess their coupling effects on each other, such as their performances and run-time when working together. To increase the TRL of the HD algorithms up to TRL 5 experimental data are needed.
Due to the fact that increasing the TRL of all the autonomous landing related algorithms is a common priority for the whole Department of Aerospace Science and Technology (DAST) at PoliMi, it has been decided to develop a facility in house.

7.2 Design variants and trade-offs

A first study has been performed considering the advantages and disadvantages of a indoor facility with respect to an outdoor one. In the last one, a flying device with the sensor assembly goes over a real relevant environment that simulates the lunar surface, whereas for an indoor facility a terrain simulator – typically smaller than the outdoor version – and the sensor assembly is moved reproducing in scale the landing maneuver. Pros and cons of the two solutions are presented in table 7.1. A second study compares the way to maneuver the sensors assembly to simulate the lander trajectory. Two solutions have been considered: a *drone* and a *robotic arm*. The first carries the navigation and HD sensors on a large scale environment, but cannot easily reproduce the lander dynamics. On the contrary, the robot allows to simulate with great precision the spacecraft dynamics through its end effector movements, but its range is much more limited with respect to the drone. The comparison between the two solution is summarized in Table 7.2. At the end of the trade-off, the preferred architecture is an indoor facility, with lander dynamics simulated by a robotic arm.

7.3 Components

To be cost effective, the selection of the components has been dictated by resources already available at PoliMi, whenever this choice

Outdoor	**Indoor**
Simplicity: no lightning system, no terrain model (+)	Complexity: terrain model and lightning system (-)
Scale more similar to a real maneuver (+)	Larger scale factor(-)
Weather and time dependent (-)	Weather and time independent (+)
Requires a large outdoor location representative of the lunar terrain (-)	No trip expenses (+)
Additional safety requirements for free flying devices in outdoor environment (-)	Total control on the experimental activity (+)
Requires wireless equipment for sensors data (-)	
Total: **-2**	Total: **+1**

Table 7.1: Feature comparison between an indoor and an outdoor facility.

would not undermine the quality of the result. The facility is therefore composed by:

- *Lunar terrain mockup*. 3D diorama manufactured in house at PoliMi through a Computer Aided Manufacturing (CAM) milling process.
- *Robotic arm*. A 7 DoF Mitsubishi PA-10 with dedicated controller to reproduce the lander dynamics real-time in the facility environment.
- *Lightning system*. Fundamental to recreate as close as possible real lunar surface light conditions.
- *Sensors assembly*. Mounted onto the end effector of the robotic arm it comprehends:
  - Camera: 8 bit grayscale frames with 1024×1024 px resolution. Field of view between 50° and 70°.
  - 6 DoF Inertial Measurement Unit (IMU).
  - Range sensor to simulate landing LASER altimeter.
- *Control PC*. Compute trajectory to be execute by the robotic arm. It simulates lander dynamics in closed-loop simulations.

Drone	**Robotic arm**
Indoor and outdoor operations (+)	Indoor only* operation (-)
Cheap (+)	Expensive (-)
Gimbaled camera to simulate attitude control (-)	Full attitude control (+)
Low precision lander dynamics reproduction (-)	High precision lander dynamics reproduction (+)
Requires wireless transmitter and receiver (-)	Wired connections (+)
High operational spatial range (+)	Low operational spatial range (-)
Total: **0**	Total: **1**

Table 7.2: Feature comparison between a drone-based and a robotic arm based facility. (*): excluding extremely high cost all-conditions robots

- *Test PC*. Algorithms under test run on this platform. In closed-loop simulations is connected to the Control PC.

7.3.1 Planetary mockup

First parameter to take into account to properly craft the lunar surface mockup is the necessary *terrain resolution*. Taking into account the beginning for the hazard detection and avoidance maneuver at about 2000 m in altitude (Sec. 1.2.2) and an envelope of the PA10 robotic arm of 1 m, it is possible to assume the *maximum scale factor equal to 2000:1*. Accuracy at touchdown is expected in the order of 10 m mainly due to navigation errors. With the assumed scale factor of 2000:1, minimum accuracy at touchdown in the simulated environment is equal to 5 mm. Since the terrain resolution should be at least one order of magnitude higher than the landing accuracy, it results a *required lunar surface mockup resolution of 0.5 mm at minimum*.
At PoliMi laboratories a numerical controlled milling machine is available with the specifications listed in Table 7.3. Some preliminary tests have been performed the best fitting material for the diorama. No particular mechanical requirements are requested to the diorama, except to not deform under its own weight. Instead, the choice is determined by optical requirements: the material must be able to be

Working area	600×1200 mm
Cutting tool	Ball nose cutter diameter: 5 mm
Position accuracy	<0.01 mm
Maximum milling depth	70 mm

Table 7.3: Milling machine characteristics

representative enough of the lunar surface optical characteristics as seen by a grayscale camera. In particular:

- The surface roughness should be lower than the required DEM resolution. If not, material imperfections would interfere with the simulation, jeopardizing fidelity and accuracy of the results due to the introduction of light diffusion and reflections. A first test performed on Styrofoam, whose granularity is in the order of 1 mm, showed these effects (Fig. 7.1);
- The density should be as low as possible in order to speed up the milling process and require a lighter and simpler support structure,
- Costs and availability from local suppliers should be optimized.

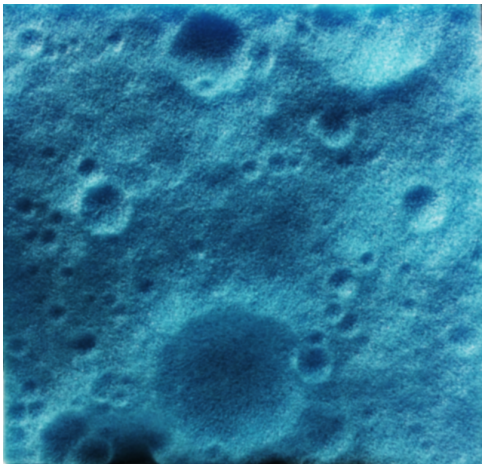


Figure 7.1: Milling test with Styrofoam. It is possible to spot the high granularity of the material.

Following these considerations, it has been selected as material for the lunar surface mockup the *RenShape® BM 5460* urethane foam characterized by the specifications in Table 7.4. In Figure 7.2a and 7.2b are displayed two different tests on the BM 5460, performed respectively with a ball cutter diameter for the milling machine of 5 mm and 12 mm and photographed in b/w.

Once a suitable material is selected, it has been proceeded with

Density	700 kg/m^3
Thermal expansion coeff.	5e-5 K^{-1}
Deflection temperature	75-80 K
Flexural strength	25-30 MPa
Compressive strength	25-30 MPa
Compressive modulus	1250-1350 MPa
Hardness	60-65 Shore D
Sheet dimensions	1500×500×100 mm

Table 7.4: RenShape® BM 5460 specifications.

(a) 5 mm cutter diameter (b) 12 mm cutter diameter

Figure 7.2: Milling tests on the RenShape® BM 5460 with a ball cutter of 5 mm and 12 mm

the selection of the DEM to be adopted in the lunar surface mockup. Some constraints drove the choice:

- operational envelope of the robotic arm;
- camera Field of View;
- milling machine working area;

- size available of RenShape sheets.

The size of the terrain model surface has been decided to be 2000×2400 mm, given by the assembly of 8 RenShape sheets measuring 1200×500 mm each. Of the 100 mm of total thickness, 70 are used for the milling machine to create the lunar terrain shape, while 30 are available to connect the mockup with the support structure beneath.
The DEM region selected reflects the following criteria:

- relevant terrain features present on the lunar surface should be included;
- real elevation data present
- artificial artifacts due to erroneous data interpolation should be avoided and eventually removed.

As stated in Chapter 4, NASA Lunar Reconnaissance Orbiter (LRO) data have been exploited to obtain reliable lunar surface informations. For the mockup surface DEM, a 14400×12000 px portion of the far side of the Moon in the GLD100 WAC Global DTM dataset have been selected. At first, a downsample to remove the DEM artifacts is performed and the resolution of the DEM is reduced to 2401×2001. Then details are added to increase resolution to 9601×8001 px by adding craters, boulders and fractal noise as described in Section 4.1. Such DEM resolution allows the mockup to be at 0.25 mm/px. In addition, depth of the DEM has been rescaled to make use of the whole vertical space available.
Discrepancies between the numerical DEM and the milling machine made mockup can be introduced with the assembly process of the 8 urethane sheets. Moreover, the cutting tool of the milling machine limits accuracy of small carved surfaces, even though the nominal precision in positioning is lower than 0.01 mm. Indeed, once crafted the diorama, it is calibrated to quantify the milling and assembling processes inaccuracies, correcting the end effector – carrying the sensors assembly – location in the computation of ground truth trajectories. The Digital Elevation Model of the physical surface of the mockup is computed during the calibration, and it must achieve the same resolution goal of 0.5 mm in the three dimensions of the numerical DEM. The calibrated DEM is then used as base onto which reconstruct ground truth trajectories in the test phase. This can be

performed with a LASER 3D scanning or with dense matching from images. The instrumentation and the know-how needed for both the techniques are already available at PoliMi. The natural color of the RenShape sheets is a light brown, but early tests reveal that no further corrections of the surface color should be required, being the images acquired in 8 bit grayscale bright enough (see Figure 7.2). Anyway, the lack of diffuse light on planets without atmosphere, even if this effect can be reproduced increasing the contrast between light and shadow through the lightning system intensity, could require the application of paint on the diorama. Further tests on attainable image contrast are required to assess such a necessity.

7.3.2 Robotic arm

A Mitsubishi PA10-7C robotic arm is available at DAST. It features 7 DoF (see Fig. 7.3) capable to handle a 10 kg payload in a operative spatial range of 1.03 meters from its shoulder joint (Fig. 7.4). Its weight amounts to 40 kg, making it easy to carry. Its kinematic

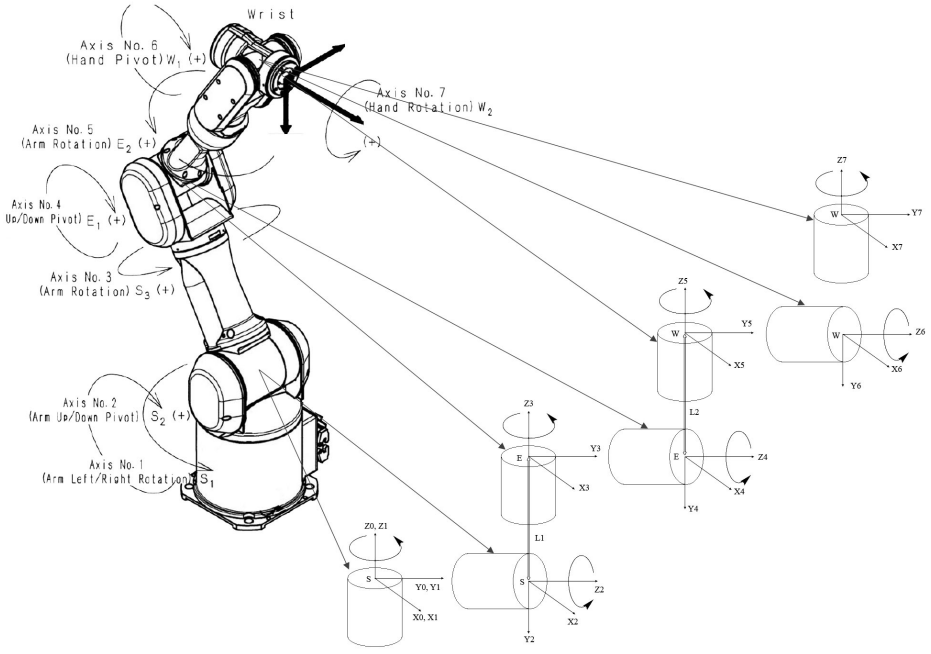


Figure 7.3: Mitsubishi PA10-7C degrees of freedom. (Credits: PA10 reference manual)

performances are:

- *Operative speed*.
  - About S1 axis (shoulder): 28.5°/s.
  - About S3 axis (elbow): 57.0°/s.
  - About W2 axis (wrist): 180°/s.
- *Position repeatability:* $\pm$0.1 mm

Two configurations have been identified, with a different relative position among the robotic arm, the lunar surface mockup and the camera. Each one has its advantages and disadvantages in terms of support structure complexity for the diorama and arm capability exploitation.

Front mounting

In this configuration diorama and robotic arm mounting base are parallel as shown in Figure 7.5. Through such configuration the arm spatial operative envelope is totally exploited and therefore the maximum surface of the diorama is reachable for. On the other hand, a reduced positioning capability of the sensor assembly on the end effector occurs due to the limited usage of the W2 (wrist) axis.
It is possible to exploit this configuration fixing the diorama to the ground, without the needs of a complex structure, or mount it perpendicular to the terrain with a supporting system as shown in Figure 7.5. The arm is positioned consequently.

Lateral mounting

Diorama and robotic arm mounting base are perpendicular one to the other as shown in Figure 7.6. In this case the maximum robotic arm end effector motional capability is exploited, due to the fact that the W2 axis regulates directly the camera inclination about the roll axis. With respect to the front mounted arm configuration, in this case it is not possible to exploit all the spatial operational envelope of the robotic arm end effector, resulting in a smaller simulation space at the same scale factor. Anyway, landing trajectories develops in a preferred direction and therefore an asymmetrical working area is not so

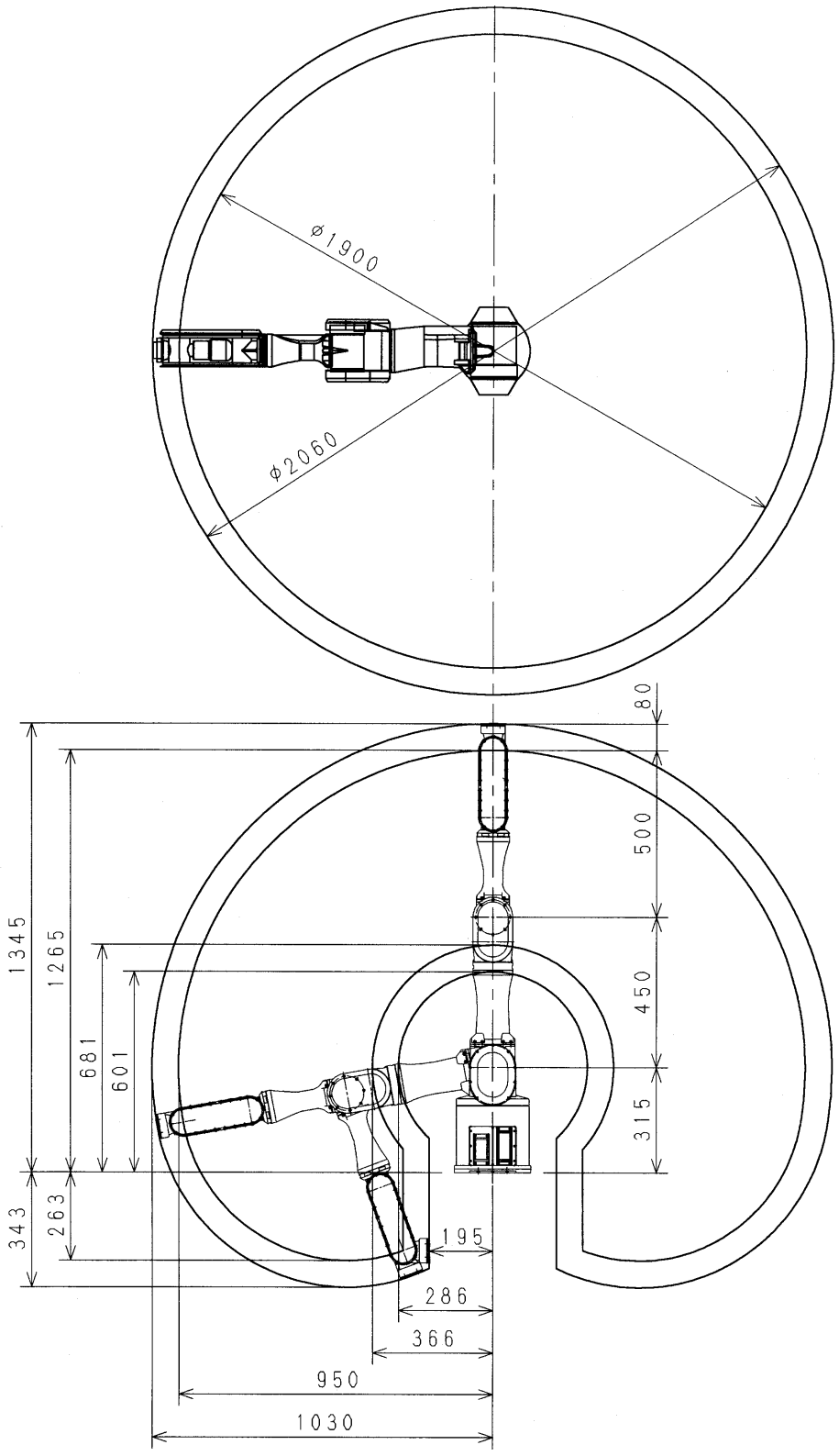


Figure 7.4: Mitsubishi PA10-7C spatial operative range. (Credits: PA10 reference manual)

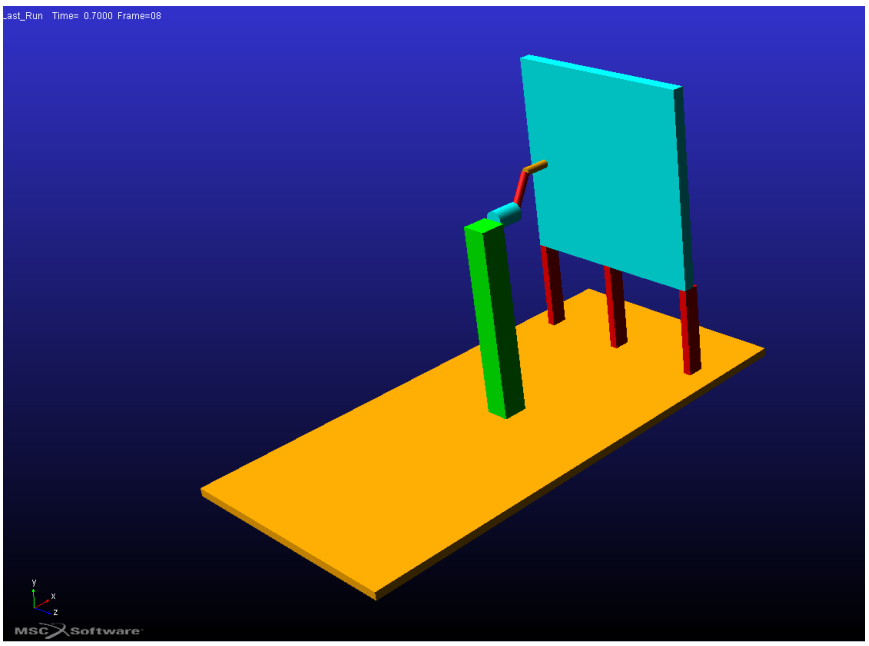


Figure 7.5: MSC Software Adams model of relative positions of robotic arm and diorama, front mounting.

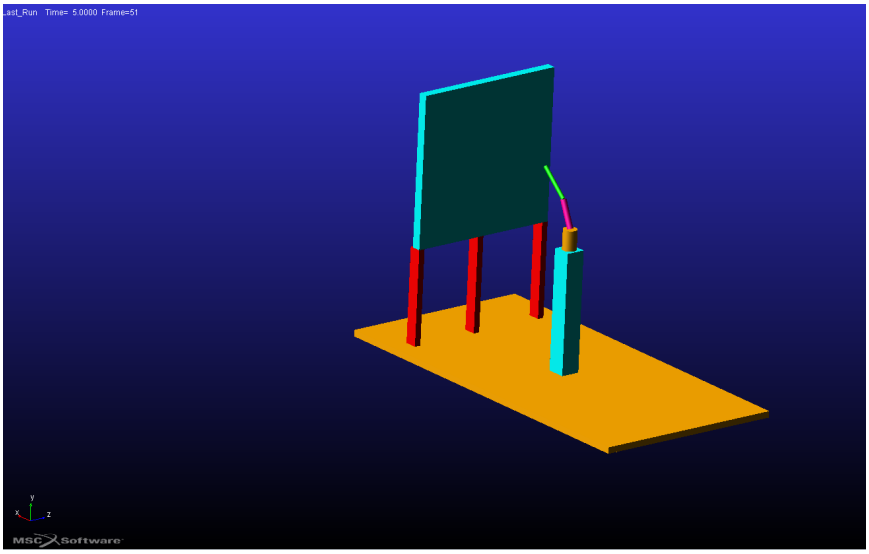


Figure 7.6: MSC Software Adams model of relative positions of robotic arm and diorama, lateral mounting.

penalizing.

Due to the fact that the use of the seventh DoF of the robotic arm allows to simulate more complex trajectories, *the lateral mounted configuration is preferred.*

7.3.3 Lightning system

It is devoted to guarantee the realistic light environment during the simulation operations. For non atmospheric celestial bodies like the Moon, diffuse lights should be avoided. At this scope, the lightning system is composed by:

- *Light source*. A 5600 K LED array with a narrow beam angle and characterized by an high Color Rendering Index (CRI). Already available in PoliMi premises, it can be adjusted according to the Sun elevation desired for the simulation.
- *Dimming system*. A non-reflective black structure dedicated to prevent external Sunlight and internal reflections to interfere with the simulation.

Theoretically, a good lightning system should provide sharp undistorted shadows on the mockup.

Chapter 8

Conclusions

An autonomous, vision based hazard detection system able to autonomously find safe landing sites during a lunar landing maneuver have been proposed in this work. It exploits an image processing algorithm to extract image features that are fed into a Feedforward Artificial Neural Network (ANN) trained with synthetic lunar images created ad hoc. The ANN outputs generates an hazard map, onto which landing sites candidates are computed taking into account landing site area, landing site hazard index, and landing site distance from the Nominal Landing Site. Tests have been performed on a different dataset with respect to the one used train the network, in order to assess performances on never seen images. A program to validate the landing sites computed has been developed, in order to assess objectively the hazard system performances in terms of True Positive, False Positive, False Negative landing sites, respectively a correctly identified, a wrongly recognized and a missed correct site. Over 90% of the landing sites identified by the system are safe, and a safe target is always selected. In the worst case encountered among the tests performed, first false positive found was ranked not worse than 3rd (average ranking: 94) allowing a backup solution in case of problems by the guidance system in computing a new landing trajectory. Moreover, execution time on a single thread of a laptop CPU of the hazard detection and landing site selection algorithms lasts only 230 ms and therefore real time compatible CPU times operations are expected using real hardware. Hence, all the requirements listed in Section 1.5 are satisfied. Promising results come also from tests of the hazard detection system on real lunar and 67P/C-G images, remarking the

system robustness on different environments and increasing the possible application of the system to all the celestial bodies without atmosphere.
A preliminary design for a facility dedicated to test and validate the hazard detection system plus navigation algorithms has been proposed: a suite of navigation sensors (a navigation camera, an IMU and a range sensor) are moved by a robotic arm over a lunar diorama, reproducing landing maneuvers in a scaled environment; a dimming system ensures a complete control over the scene illumination, providing realistic lighting conditions.

8.1 Future developments

For what concerns the hazard detection system, many options can be walked through in order to try to enhance the performances:

- extension of the images dataset to include different altitudes in order to make the artificial network able to discriminate between a boulder and a small stone depending on the altitude;
- investigation of an adaptive hazard threshold depending on the Sun elevation angle in order to further optimize the HD system performances;
- implementation of other machine learning items, such as cascade neural networks, or deep learning techniques;
- refining of training datasets with the inclusion of more accurate renderings;
- investigation of other image processing technique to provide still not tried inputs to the neural network;

The facility, which is currently at the detailed design stage, has the ultimate goal to provide the testbed for open-loop and closed-loop simulations not only of navigation and hazard detection, but in general visual based autonomous systems.

Bibliography

[1] eoPortal Directory. https://directory.eoportal.org/web/eoportal/satellite-missions/c-missions/chang-e-3. Last visit: 10/06/2015.

[2] Esa plans for lunar exploration. Technical report, European Space Agency, 2014.

[3] Indian Space Projects, Chandrayaan-2. http://isp.justthe80.com/moon-exploration/chandrayaan—2. Last visit: 10/06/2015.

[4] Mars Science Laboatory, guided entry. http://mars.jpl.nasa.gov/msl/mission/technology/insituexploration/edl. Last visit: 10/06/2015.

[5] European Space Agency. *ExoMars 2018 Landing Site Selection User Manual*, December 2013.

[6] S. Barraclough, A. Ratcliffe, R. Buchwald, H. Scheer, M. Chapuy, M. Garland, and D. Rebuffat. Phootprint: A European Phobos Sample Return Mission. *LPI Contributions*, June 2014.

[7] OSIRIS-REx website. http://www.asteroidmission.org/. Last visit: 10/06/2015.

[8] Rosetta official website. http://rosetta.esa.int. Last visit: 11/06/2015.

[9] Alex Foessel and William (Red) L. Whittaker. Mmw-scanning radar for descent guidance and landing safeguard. In Canadian Space Agency, editor, *Proceedings 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, June 2001.

[10] Farzin Amzajerdian, Michael Vanek, Larry Petway, Diego Pierrottet, George Busch, and Alexander Bulyshev. Utilization of 3-d imaging flash lidar technology for autonomous safe landing on planetary bodies. In *Proc. of SPIE Vol*, volume 7608, pages 760828–1.

[11] Farzin Amzajerdian, Diego F Pierottet, Larry B Petway, and Michael D Vanek. Development of lidar sensor systems for autonomous safe landing on planetary bodies. 2010.

[12] DragonEye 3D Flash LIDAR Space Camera. http://www.advancedscientificconcepts.com/products/dragoneye.html. Advanced Scientific Concepts, Inc.

[13] Sinplex.eu. http://www.sinplex.eu. Accessed: 2015-05-05.

[14] Stephen R. Steffes. Small integrated navigator for planetary exploration (sinplex). In *ESA Workshop on Avionics Data, Control and Software Systems (ADCSS)*, 2012.

[15] Stephen R. Steffes. Development and analysis of shefex-2 hybrid navigation system experiment. Master's thesis, DLR - Bremen, 2013.

[16] alhat.jpl.nasa.gov. http://alhat.jpl.nasa.gov/. Accessed: 2015-05-06.

[17] Stephen Paschall II, Tim Brady, Babak E. Cohanim, and Ronald R. Sostaric. A self contained method for safe and precise lunar landing. In *2008 IEEE Aerospace Conference*, number 1643. Institute of Electrical and Electronics Engineers, March 2008.

[18] C.D. Epp and T.B. Smith. Autonomous precision landing and hazard detection and avoidance technology (alhat). In *Aerospace Conference, 2007 IEEE*, pages 1–7. Institute of Electrical and Electronics Engineers, March 2007.

[19] Thomas Voirin, Jeff Delaune, G Le Besnerais, JL Farges, Clément Bourdarias, and Hans Krueger. Challenges of pinpoint landing for planetary exploration : the lion absolute vision-based navigation system step-wise validation approach. In *10th International Planetary Probe Workshop*, 2013.

[20] Jeff Delaune. *Navigation visuelle pour l'atterrissage planétaire de précision indépendante du relief*. PhD thesis, 2013. Thèse de doctorat dirigée par Le Besnerais, Guy et Farges, Jean-Loup Systèmes embarqués et robotique Toulouse, ISAE 2013.

[21] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *Int. J. Comput. Vision*, 65(1-2):43–72, November 2005.

[22] Andres Huertas, Yang Cheng, and Richard Madison. Passive imaging based multi-cue hazard detection for spacecraft safe landing. In *Aerospace Conference, 2006 IEEE*, pages 14–pp. IEEE.

[23] Eleanor S. Crane. Vision-based hazard estimation during autonomous lunar landing. Master's thesis, Stanford University, Stanford, CA, 06/2014 2014.

[24] Kyoung Mu Lee and C.-C.Jay Kuo. Shape from shading with a generalized reflectance map model. *Computer Vision and Image Understanding*, 67(2):143 – 160, 1997.

[25] Berthold K.P. Horn. Understanding image intensities. *Artificial Intelligence*, 8(2):201 – 231, 1977.

[26] B. Parreira, E. Di Sotto, A. Caramagno, and J. Rebordão. Hazard avoidance for planetary landing: Gnc design and performance assessment. In *7th International ESA Conference on Guidance, Navigation & Control Systems*. European Space Agency, 2008.

[27] M. Bajracharya. Single image based hazard detection for a planetary lander. In *Automation Congress, 2002 Proceedings of the 5th Biannual World*, volume 14, pages 585–590, 2002.

[28] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. In *Computer Vision and Pattern Recognition, 2004.*, volume 1, pages I–652–I–659 Vol.1, June 2004.

[29] A Wedel, U. Franke, J. Klappstein, T. Brox, and D. Cremers. Realtime Depth Estimation and Obstacle Detection from Monocular Video. 2006.

[30] N. Trawny, J. M. Carson, A. Huertas, M. E. Luna, V. E. Robak, A. E. Johnson, K. E. Martin, and C. Y. Villalpando. Helicopter flight testing of a real-time hazard detection system for safe lunar landing. In *SPACE Conference and Exposition*. American Institute of Aeronautics and Astronautics, 2013.

[31] L. Matthies, A. Huertas, Y. Cheng, and A. Johnson. Landing hazard detection with stereo vision and shadow analysis. In *Infotech@Aerospace Conference and Exhibit*. American Institute of Aeronautics and Astronautics, 2007.

[32] Andrew Johnson and Tonislav Ivanov. Analysis and testing of a lidar-based approach to terrain relative navigation for precise lunar landing. 2011.

[33] Andrew E Johnson, Allan R Klumpp, James B Collier, and Aron A Wolf. Lidar-based hazard avoidance for safe landing on mars. *Journal of guidance, control, and dynamics*, 25(6):1091–1099, 2002.

[34] Hans Krüger and Stephan Theil. Tron-hardware-in-the-loop test facility for lunar descent and landing optical navigation.

[35] Marco Sagliano, Hans Krüger, and Stephan Theil. Tron tool: Representing a moon landing scenario in tron.

[36] Piergiorgio Lanza, Nicoletta Noceti, Corrado Maddaleno, Antonio Toma, Luca Zini, and Francesca Odone. A vision-based navigation facility for planetary entry descent landing. In Andrea Fusiello, Vittorio Murino, and Rita Cucchiara, editors, *ECCV Workshops (2)*, volume 7584 of *Lecture Notes in Computer Science*, pages 546–555. Springer, 2012.

[37] G Flandin, B Polle, B Frapard, P Vidal, C Philippe, and T Voirin. Vision based navigation for planetary exploration.

[38] HOMER testbed, Airbus Defence and Space. http://www.space-airbusds.com/en/news2/airbus-defence-and-space-s-homer-demonstrator.html. Last visit: 25/05/2015.

[39] ALHAT news. http://alhat.jpl.nasa.gov/news.php. Last visit: 25/05/2015.

[40] G. Bradski. *Dr. Dobb's Journal of Software Tools*, 2000.

[41] S. Nissen. Implementation of a Fast Artificial Neural Network Library (fann). *Report, Department of Computer Science University of Copenhagen (DIKU)*, 31, 2003.

[42] S. M. Platek, J. P. Keenan, and T. K. Shackelford. Evolutionary Cognitive Neuroscience - MIT Press, Cambridge, 2007, 620 pp. *Minds and Machines*, 19(2):275–278, 2009.

[43] G.M. Shepherd. *The Synaptic Organization of the Brain*. Oxford University Press, Oxford, 1990.

[44] S. O. Haykin. *Neural Networks and Learning Machines, 3rd ed.* Pearson, 2008.

[45] W. J. Freeman. *Mass Action in the Nervous System*. Academic Press, 1975.

[46] Santiago Ramón y Cajal. *Histologie du système nerveux de l'homme & des vertèbrès.*, volume v. 1. Paris Maloine, 1911.

[47] Frank Rosenblatt. The perceptron–a perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory, 1957.

[48] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA, 1969.

[49] Stephen Grossberg. Contour enhancement, short term memory, and constancies in reverberating neural networks. In *Studies in Applied Mathematics*. MIT, Boston, 1973.

[50] P. T. Churchland and T. J. Sejnowski. *The Computational Brain*. Computational Neuroscience, 1992.

[51] R. Royas. *Neural Networks: A Systematic Introduction*. Springer, 1996.

[52] Gail A. Carpenter and Stephen Grossberg. The art of adaptive pattern recognition by a self-organizing neural network. *Computer*, 21(3):77–88, 1988.

[53] P. Kerlirzin and F. Vallet. Robustness in Multilayer Perceptrons. 1993.

[54] Olivier Bousquet Ulrike von Luxburg and Gunnar Rätsch. Advanced lectures on machine learning. 2004.

[55] R. Beale and T. Jackson. *Neural Computing: An Introduction*. IOP Publishing Ltd., Bristol, UK, UK, 1990.

[56] Robert Andrew Wilson and Frank C Keil. *The MIT encyclopedia of the cognitive sciences*. 2001.

[57] Neural Networks and Deep Learning, online free book. http://neuralnetworksanddeeplearning.com/. Accessed: 2015-05-10.

[58] Martin Riedmiller and I. Rprop. Rprop - description and implementation details, 1994.

[59] C. Igel and M. Hüsken. Improving the Rprop Learning Algorithm. In *Proceedings of the Second International Symposium on Neural Computation, NC 2000*, pages 115,121. ICSC Academic Press, 2000.

[60] Hao Yu and B.M. Wilamowski. Levenberg-marquardt training.

[61] M.I.A. Lourakis and A.A. Argyros. Is levenberg-marquardt the most efficient optimization algorithm for implementing bundle adjustment? In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1526–1531 Vol. 2, Oct 2005.

[62] Martin F. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *NEURAL NETWORKS*, 6(4):525–533, 1993.

[63] Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 4. Springer New York.

[64] S. J. D. Prince. *Computer Vision: Models, Learning, and Inference*. Cambridge University Press.

[65] Y. LeCun, L. Bottou, G. Orr, and K. Muller. Efficient BackProp. In G. Orr and Muller K., editors, *Neural Networks: Tricks of the trade*. Springer, 1998.

[66] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69, 1982.

[67] Alessio Emilio Colombo. Sonde robotiche per l'atterraggio su superfici planetarie : supporto all'atterraggio morbido mediante generazione di mappe di pericolosità. Master's thesis, Politecnico di Milano, December 2012.

[68] Simone Bernardi. Determinazione Mediante Reti Neurali di Mappe di Pericolosità in Fase di Atterraggio su Corpi Celesti. Master's thesis, Politecnico di Milano, 2002.

[69] LROC RDR Product Select. http://wms.lroc.asu.edu/lroc/rdr_product_select. Accessed: 2015-04-10.

[70] P. Lunghi and M. Lavagna. Autonomous Vision-based Hazard Map Generator for Planetary Landing Phases. In *65 th International Astronautical Congress*. International Astronautical Federation, 2014.

[71] CA Cross and DL Fisher. The computer simulation of lunar craters. *Monthly Notices of the Royal Astronomical Society*, 139(2):261–272, 1968.

[72] UJ Shankar, Wen-Jong Shyong, TB Criss, and D Adams. Lunar terrain surface modeling for the alhat program. In *Aerospace Conference, 2008 IEEE*, pages 1–10. IEEE, 2008.

[73] Alain Fournier, Don Fussell, and Loren Carpenter. Computer rendering of stochastic models. *Commun. ACM*, 25(6):371–384, June 1982.

[74] F. Hörz, R. Grieve, G. Heiken, P. Spudis, and A. Binder. Lunar Surface Processes. 1991.

[75] Persistence of Vision (TM) Raytracer. http://www.povray.org/. Persistence of Vision Pty. Ltd. (2004), Victoria, Australia.

[76] M. S. Nixon and A. S. Aguado. *Feature Extraction and Image Processing*. Newnes, 2002.

[77] Mubarak Shah. Fundamentals of Computer Vision. 1997.

[78] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[79] Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. Learning depth from single monocular images. In *Advances in Neural Information Processing Systems*, pages 1161–1168, 2005.

[80] J.M.S. Prewitt. Object enhancement and extraction. *Picture processing and Psychopictorics*, 1970.

[81] H. B. Kekre and S. M. Gharge. Image Segmentation using Extended Edge Operator for Mammographic Images. *International Journal on Computer Science and Engineering*, 2(4):1086–1091, 2010.

[82] E Hildreth. Theory of edge detection. *Proceedings of Royal Society of London*, 207(187-217):9, 1980.

[83] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986.

[84] How It Works. Canny edge detector.

[85] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons, 2001.

[86] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.

[87] GperfTools Website. https://code.google.com/p/gperftools/. Last visit: 25/06/2015.

[88] Linux User's Manual. http://man7.org/linux/man-pages/man1/time.1.html. Last visit: 25/06/2015.

[89] www.byclb.com, Neural Networks tutorial. www.byclb.com/TR/Tutorials/neural_networks/ch10_1.htm. Accessed: 2015-05-02.

[90] Thomas P Vogl, JK Mangis, AK Rigler, WT Zink, and DL Alkon. Accelerating the convergence of the back-propagation method. *Biological cybernetics*, 59(4-5):257–263, 1988.

Appendix A

Activation functions

Heaviside function

Also known as *threshold* function, Heaviside's maps the input to a binary output, as shown in Fig. A.1:

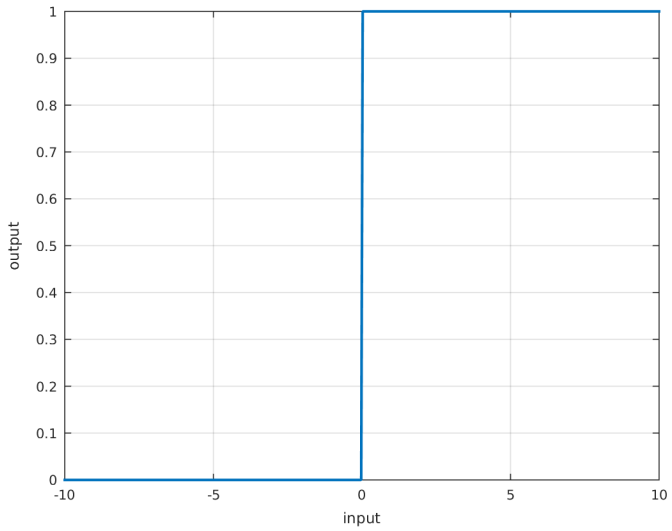


Figure A.1: Heaviside activation function.

$$\mathcal{H}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

Linear function

Linear proportionality between input and output. Due to the fact that it is not bounded, also weights and and neuron output will not be. This leads to much longer training times with respect to bounded activation functions[89]. Its definition is of course:

$$y = ax \tag{A.1}$$

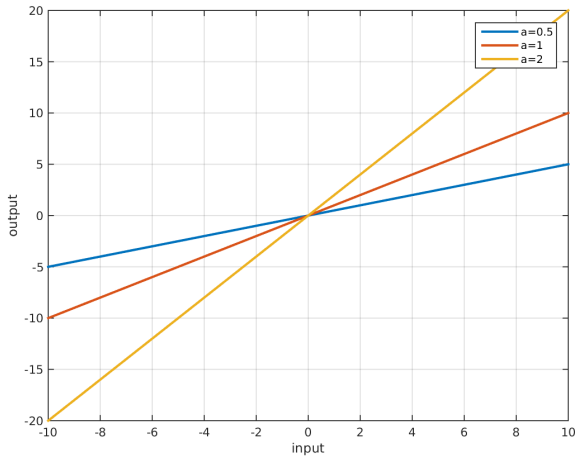


Figure A.2: Linear activation function.

where a is a real constant.

Ramp

Linear function in a specified domain. Useful to not saturate the neuron output as a pure linear function does (see Sec. A). In the presented version, it is centered in the origin (0,0) both in input and output axis to provide a odd function.

It may be written as:

$$\mathcal{R}(x) = \begin{cases} -1 & \text{if } \mathcal{R}(x) < -1 \\ ax & \text{if } -1 \leq \mathcal{R}(x) \leq 1 \\ 1 & \text{if } \mathcal{R}(x) > 1 \end{cases}$$

where a is a real constant.

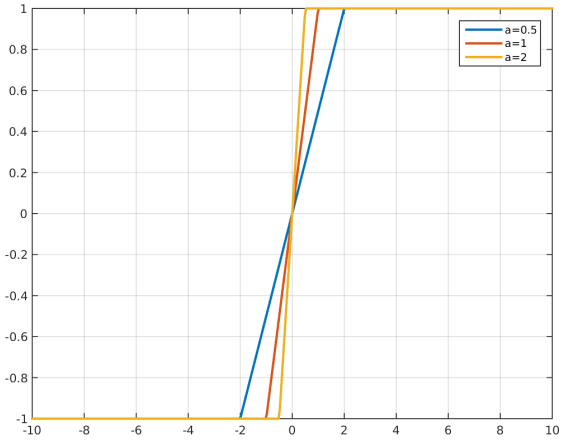


Figure A.3: Ramp activation function.

Logistic sigmoid

Bounded between [0,1], it is one of the most used activation functions. It is monotonically increasing, and acts as a threshold function for inputs approaching plus or minus infinity. It is differentiable, thus it is exploitable in many efficient training algorithms. It is defined as:

$$logsig(x) = \frac{1}{1 + e^{-ax}} \tag{A.2}$$

where a is a real constant.

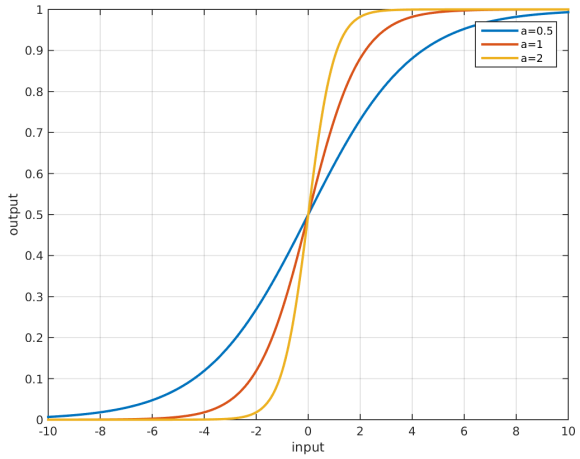


Figure A.4: Logistic sigmoid activation function.

Hyperbolic tangent

Another kind of sigmoid function, as popular as the logistic sigmoid. Bounded between [-1,1], it is computationally more efficient with respect to the logistic sigmoid[65].

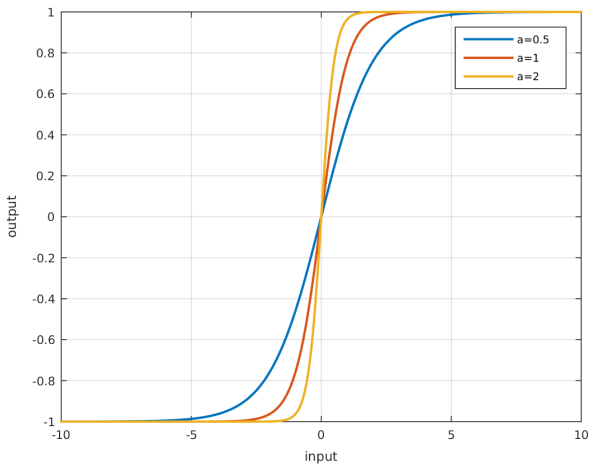


Figure A.5: Hyperbolic tangent activation function.

The hyperbolic tangent function appears as:

$$\tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{A.3}$$

but in the programs developed in this thesis, it has been implemented an equivalent version [90], that is faster to compute.

$$tansig(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{A.4}$$

Appendix B

Test dataset

The test dataset is composed by 8 doublets of lunar surface image and relative hazard map ground truth. All the images have been generated through LROC data, adding fractal noise and features to increase resolution, as described in Chapter 4. The whole test dataset has the following characteristics:

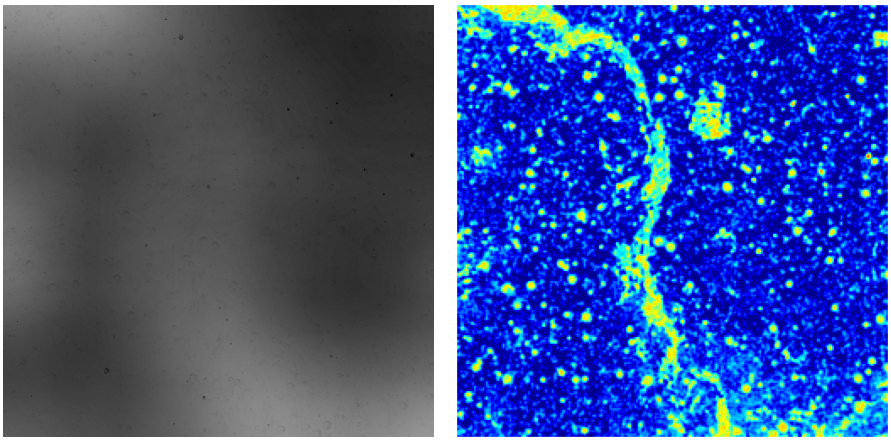
Table B.1: Test dataset characteristics

Altitude	2000 m
Camera attitude	vertical
Sun inclination	15° or 80°
Sun azimuth	15°

Hazard maps are represented through the colormap in Figure B.1, that sweeps from blue (safe) to red (unsafe).

Figure B.1: Colormap adopted for the hazard map graphical representation.

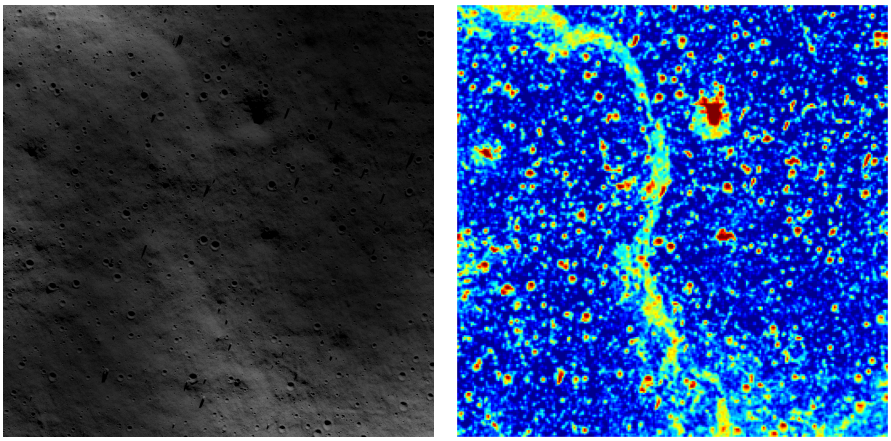
Test image 1



(a) DEM image rendering (b) Hazard map

Figure B.2: Test image 1, 80° Sun inclination angle.

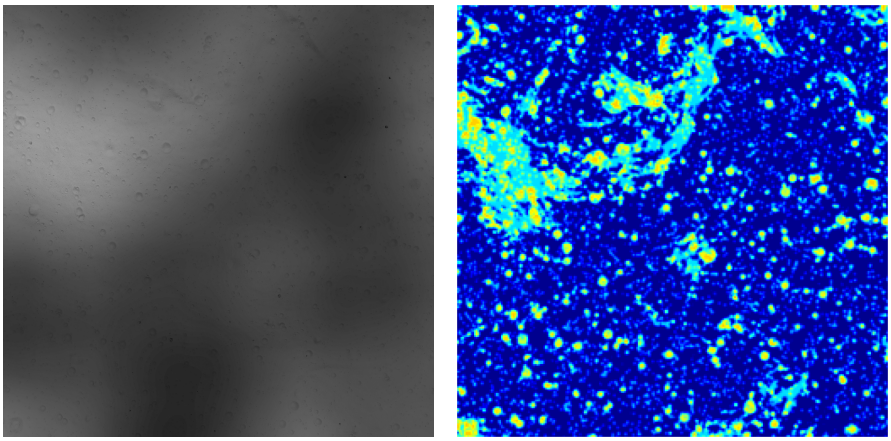
Test image 2



(a) DEM image rendering (b) Hazard map

Figure B.3: Test image 2, 15° Sun inclination angle.

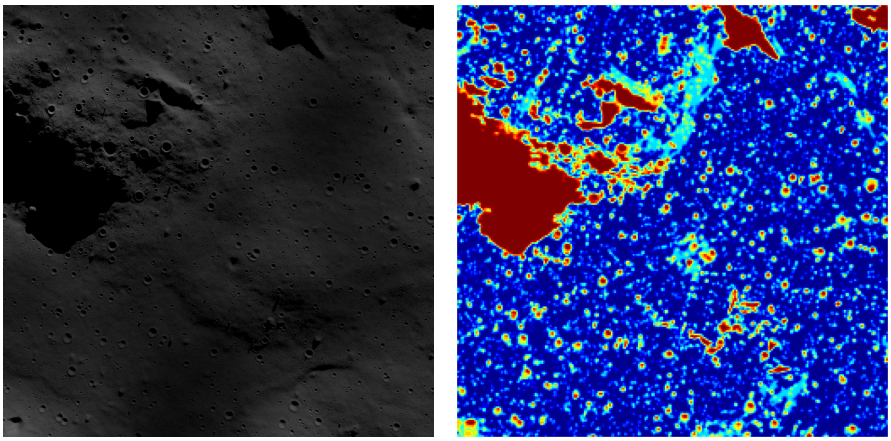
Test image 3



(a) DEM image rendering (b) Hazard map

Figure B.4: Test image 3, 80° Sun inclination angle.

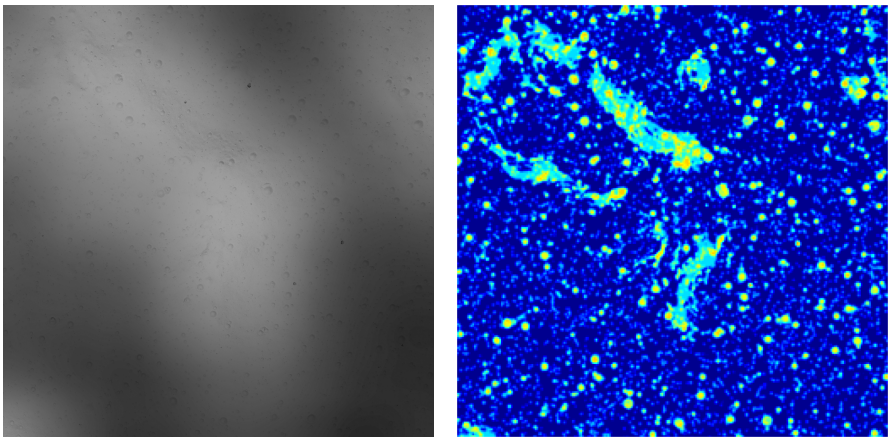
Test image 4



(a) DEM image rendering (b) Hazard map

Figure B.5: Test image 4, 15° Sun inclination angle.

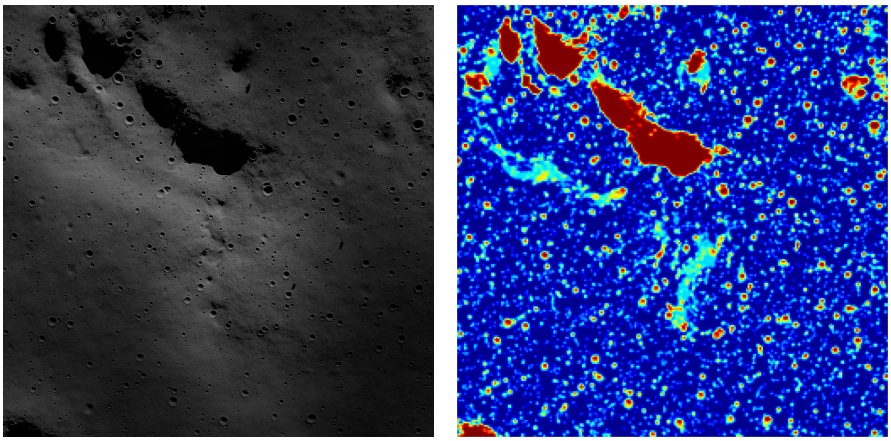
Test image 5



(a) DEM image rendering (b) Hazard map

Figure B.6: Test image 5, 80° Sun inclination angle.

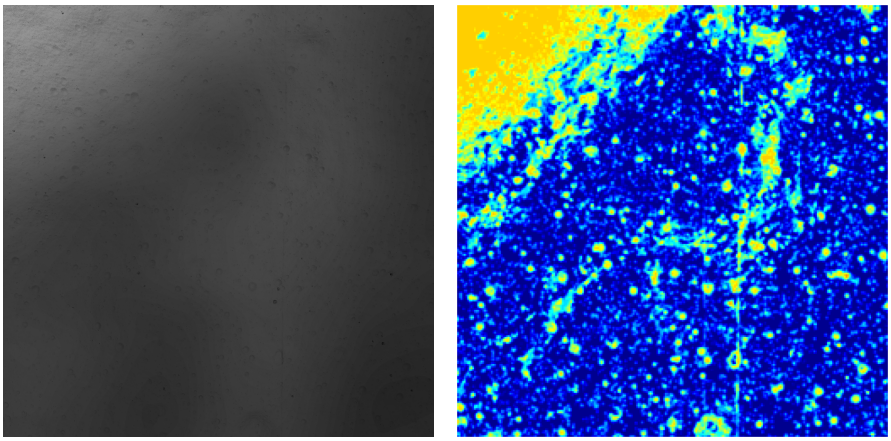
Test image 6



(a) DEM image rendering (b) Hazard map

Figure B.7: Test image 6, 15° Sun inclination angle.

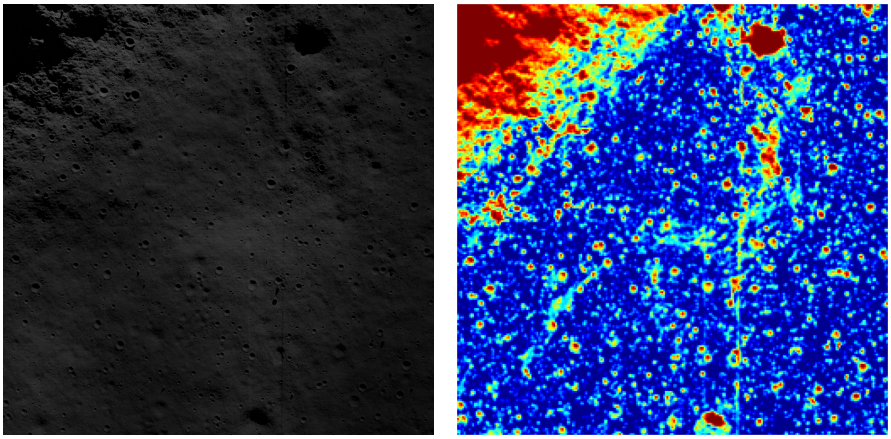
Test image 7



(a) DEM image rendering (b) Hazard map

Figure B.8: Test image 7, 80° Sun inclination angle.

Test image 8



(a) DEM image rendering (b) Hazard map

Figure B.9: Test image 8, 15° Sun inclination angle.