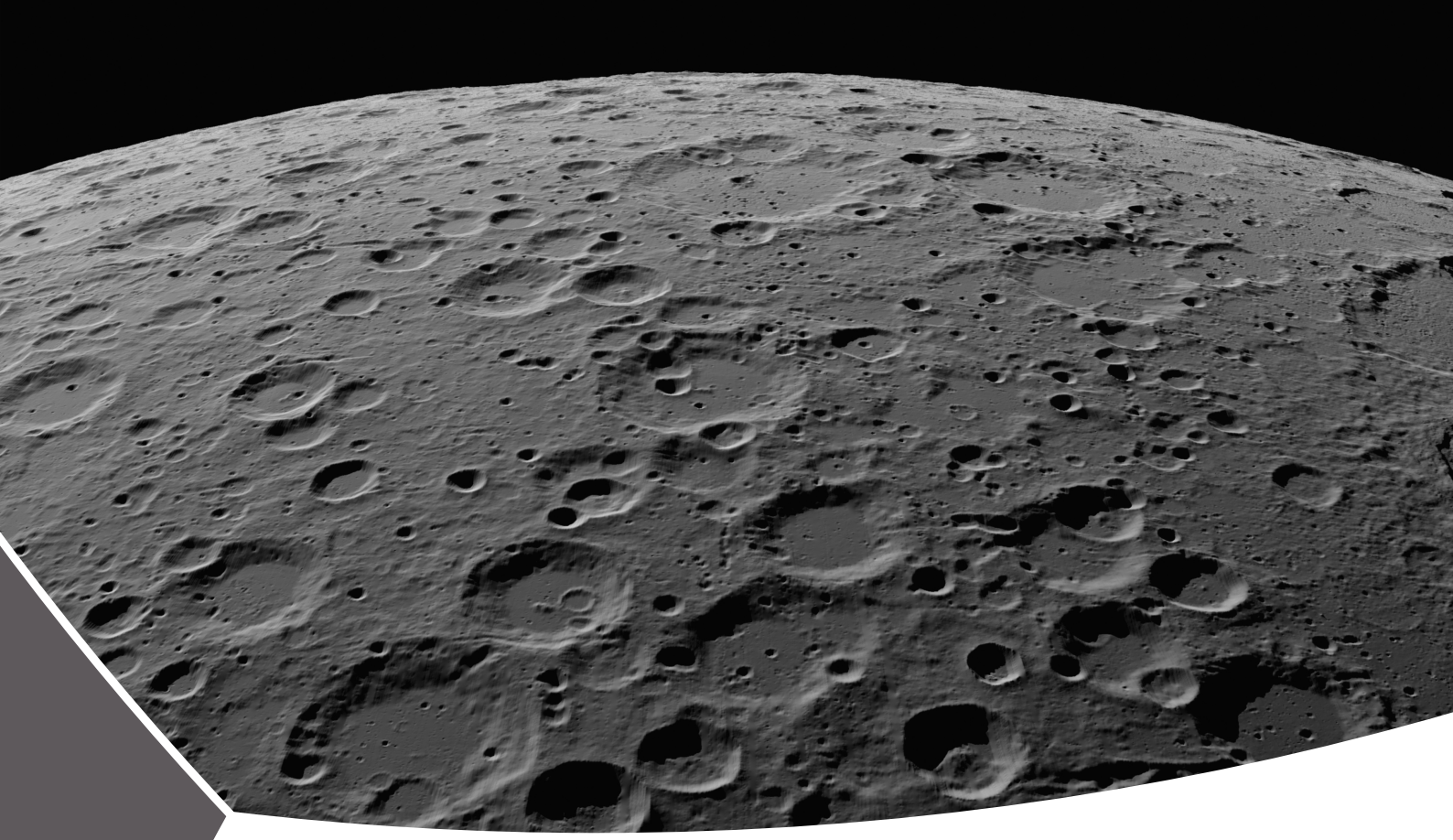


Project

ATON

Autonomous Terrain-based
Optical Navigation

Final Report

DLR

Publisher **Deutsches Zentrum**
für Luft- und Raumfahrt e.V.
German Aerospace Center

Institute of Space Systems
Department of Guidance, Navigation and Control Systems

Editor Dr.-Ing. Stephan Theil

Address Robert-Hooke-Str. 7
28359 Bremen

Authors N. Ammann (BS FT-ULF), Dr.-Ing. F. Andert (BS FT-ULF), T. Franz (BS SC-SRV), H. Krüger (HB RY-GNC), M. Lingenauber (OP RM-PEK), D. Lüdtke (BS SC-SRV), Dr.-Ing. B. Maass (HB RY-GNC), C. Paproth (BA OS-WIN), Dr.-Ing. S. Theil (HB RY-GNC), Dr. rer. nat. J. Wohlfeil (BA OS-EDP)

Published Bremen, April 2018

ISRN DLR-FB-2018-45

gnc.DLR.de

Project

ATON

Autonomous Terrain-based
Optical Navigation

Final Report

ISRN DLR-FB-2018-45

Table of Contents

Preface

ATON stands for **A**utonomous **T**errain-based **O**ptical **N**avigation — a project established and implemented by the German Aerospace Center (DLR). The project goal is to research, develop and test new navigation techniques and technologies enabling precise and safe landing on extraterrestrial solar system bodies.

This document is a status report of the project ATON. Chap. 1 introduces the details of the project. Motivation and goals are outlined. Furthermore, the state of the art is reflected by referencing previous and similar ongoing activities.

In Chap. 2 first historic and currently developed missions are reviewed in order to select a reference mission profile, trajectory and requirements for the development of technologies within the project ATON. An overview of the system design and its architecture is provided.

The following Chap. 3 describes the techniques and technologies developed and implemented within the project. The functional principles of the different modules of the developed ATON system are presented.

Throughout the project ATON several tests have been conducted to test functions and performance of the technologies at different implementation levels. Chap. 4 describes for each achieved milestone the setup as well as the results.

The following Chap. 5 summarizes the lessons learned from the project.

The report closes with Chap. 6 providing conclusions and outlook.

List of Acronyms

ADAPT	Autonomous Descent and Ascent Powered-Flight Testbed
ALHAT	Autonomous Landing and Hazard Avoidance Technology
alt	altitude
ATON	Autonomous Terrain-based Optical Navigation
AutoNav	Autonomous Vision-based Orbit Navigation
BRDF	Bidirectional Reflectance Distribution Function
BSM	Binary Shadow Matching
CAM	Navigation Camera
CRS	Camera reference system
CNES	Centre National d'Études Spatiales
COBALT	CoOperative Blending of Autonomous Landing Technologies
CPU	Central Processing Unit
cr	cross-range
DEM	Digital Elevation Model
DLR	German Aerospace Center
DO	Descent Orbit
DOF	Degree of Freedom
DOI	Descent Orbit Injection
dr	down-range
DSN	Deep Space Network
ECEF	Earth-Centered Earth-Fixed
ECI	Earth-Centered Inertial
EKF	Extended Kalman Filter
EPnP	Efficient Perspective-n-Point
ESA	European Space Agency
FIFO	First-In, First-Out
FPGA	Field-Programmable Gate Array
FOG	Fiber Optical Gyroscope
FOV	Field of View
G-FOLD	Guidance algorithm for Fuel Optimal Large Diverts
GNC	Guidance, Navigation and Control
GNSS	Global Navigation Satellite System
GPIO	General Purpose Input/Output
GPS	Global Positioning System
GRAIL	Gravity Recovery and Interior Laboratory
GSD	Ground Sampling Distance
GUI	Graphical User Interface
HDA	Hazard Detection and Avoidance
HiL	Hardware-in-the-Loop
HRN	Hazard Relative Navigation
ICD	Interface Control Document
ICP	Iterative Closest Point
I/F	Interface
IP	Internet Protocol
INSIGHT	Interior Exploration using Seismic Investigations, Geodesy and Heat Transport
IMU	Inertial Measurement Unit
JAXA	Japanese Aerospace Exploration Agency
JPL	Jet Propulsion Laboratory
KLT	Lukas Tomasi Kanade
LA	Laser Altimeter
LAN	Local Area Network
LED	light-emitting diode
LIDAR	Light Detection and Ranging
LM	Lunar Module
LOS	Line of Sight
LTRF	Laser Tracker reference system
MASCOT	Mobile Asteroid Surface Scout
MCMF	Moon-Centered Moon-Fixed
MDSD	Model-Driven Software Development
MEMS	Microelectromechanical systems
MiL	Model-in-the-Loop
MOF	Meta-Object Facility
MPE	Maximum permissible error
NASA	National Aeronautics and Space Administration
NEXT	Next Exploration Science and Technology Mission
NIS	Normalized Innovations Squared
NPAL	Navigation for Planetary Approach and Landing
PCA	Principal Component Analysis
PD	Powered Descent
PDI	Powered Descent Initiate

PnP	Perspective-n-Point
PiL	Processor-in-the-Loop
PILOT	Precise Intelligent Landing using On-Board Technology
QCP	quaternion-based characteristic polynomial
RANSAC	Random Sample Consensus
RLG	Ring Laser Gyroscope
RMS	Root Mean Square
RTEMS	Real-Time Executive for Multiprocessor Systems
ROI	Regions of Interest
SAPOS	Satellitenpositionierungsdienst der deutschen Landesvermessung
SC	spacecraft
SCRF	spacecraft reference system
SGM	Semi-Global Matching
SiL	Software-in-the-Loop
SLAM	Simultaneous Localization and Mapping
STR	Star Tracker
SysML	System Modeling Language
TCP	Tool Center Point
TMRF	Terrain Model reference system
TRL	Technology Readiness Level
TRN	Terrain Relative Navigation
TRON	Testbed for Robotic Optical Navigation
UDP	User Datagram Protocol
UKF	Unscented Kalman Filter
UML	Unified Modeling Language
UTC	Universal Time Coordinated
VBNAT	Vision-Based Navigation Analysis Tool
VisNav	Vision-based Navigation
VTB	Vertical Testbed
XML	Extensible Markup Language

1 Introduction

1.1 Motivation

Upcoming space exploration missions envisage precise and safe landing on planetary bodies, e.g. to land at fuel depots, outposts, on specific spots for investigating local phenomena, or for collecting a sample return canister. Such missions will be performed at various distances from Earth, reaching out from the vicinity of Earth and the Moon, targeting Mars, asteroids or the icy moons of Jupiter and Saturn.

One of the most essential prerequisites for mission success is the knowledge of the spacecraft's position, attitude, velocity, and rotational speed with respect to the target body's reference system. Currently, the available technology does not provide this information with sufficient accuracy and timing. Even though Earth-based tracking delivers position and speed, there are several problems connected with that method. With growing distance the accuracy becomes degraded and a prohibitive delay is introduced. Furthermore this method can lead to a potential blocking of the line of sight to the spacecraft by the target. Current on-board sensors which may contribute to the knowledge of the desired states are altimeters and velocimeters. While delivering measurements relative to the underlying surface they either lack the explicit context to the target body's reference system at all, or they require prohibitively complex models of the target geometry.

Consequently a considerable demand has evolved for two capabilities. One allows for the building of sensors which deliver position and attitude relative to the target's reference system to fill the measurement gap mentioned above. Camera-based sensors and methods for processing their associated images are very promising means for providing this capability. Moreover, additionally to the pose, such sensors can provide information about the surface relative speed. The second desired capability is to build a navigation system which combines these camera-based sensors with traditional sensors and methods for finally enabling the accomplishment of the challenges described above.

1.2 Goals

The German Aerospace Center (DLR) has been active in the field of planetary science for decades and has been involved in many interplanetary missions providing instruments and technologies. Technologies for landing have been developed for the lander Philae of the Rosetta mission which landed November 12, 2014 on the comet 67P/Churyumov–Gerasimenko ten years and eight months after departing Earth [73, 91]. Similarly, the asteroid landing package Mobile Asteroid Surface Scout (MASCOT) was developed by DLR in cooperation with the Japanese Aerospace Exploration Agency (JAXA) and the French Centre National d'Études Spatiales (CNES) [32]. It is traveling onboard JAXA's mission Hayabusa 2 to its landing target asteroid 162173 Ryugu. The landing is foreseen for October 2018.

As an evolution of the landing technologies which were applied to Philae and MASCOT, technologies for precise and safe landing are in the focus of DLR's research and development activities. One element is the project ATON. The project was initiated in 2010 and started from several already available technologies in the domain of image processing, optical navigation and state estimation.

The overarching goal of ATON is the development and demonstration of an optical navigation system for exploration missions and its associated technologies which enable an autonomous, precise and safe landing on a celestial body. The goals of the project are:

- Development of a flexible system concept allowing tailored solutions for different missions,
- Development of image processing and optical navigation techniques for absolute and relative navigation,
- Development of navigation filtering techniques fusing all available sensor data and image processing outputs,
- Verification of all algorithms implemented as software in Model-in-the-Loop (MiL), Software-in-the-Loop (SiL), Processor-in-the-Loop (PiL) and Hardware-in-the-Loop (HiL) setups including the development of software and hardware tools for realistic simulations,
- Verification of the navigation system performance in open-loop and closed-loop, and
- In-flight demonstration of the navigation system in terrestrial test environments.

ATON was set up as a technology research project without a concrete mission to be served. This provided more degrees of freedom than in mission-driven developments and it allowed the possibility of exploring different approaches to the same problem in parallel, to initiate and explore new ideas, and to investigate different solutions more thoroughly. One of the main differences from many agency-driven technology developments in the same area is that all elements of the optical navigation system were continuously researched and developed by the same enterprise and the same team. This gave the team a broader view of the challenges and the chance to gain a deeper understanding of the optical navigation system and the underlying principles.

1.3 State of the Art

When the project ATON was initiated in 2010 current activities were NASA's Autonomous Landing and Hazard Avoidance Technology (ALHAT) program, JAXA's lunar and asteroid exploration program as well as activities in Europe as European Space Agency (ESA)'s Lunar Lander and projects of ESA's Aurora program like Autonomous Vision-based Orbit Navigation (AutoNav), Vision-based Navigation (VisNav), Navigation for Planetary Approach and Landing (NPAL), and Vision-Based Navigation Analysis Tool (VBNAT). ATON's objectives have meshed very well with these activities in the past, with the creation of complementary research and development in the field of optical navigation for planetary exploration. In the following the parallel activities of other agencies are summarized.

National Aeronautics and Space Administration (NASA)
In 2006 the ALHAT frame was initiated [36] and set the goal of developing new Guidance, Navigation and Control (GNC) technologies in sensors and algorithms for crewed or robotic landers that enable autonomous and safe precision landing onto the lunar surface under any terrain lighting conditions. In hardware terms, these new technologies are entirely either passive or active optical sensors. Over time research in the United States of America widened to apply optical navigation techniques for essentially any exploration mission to solid bodies in the solar system including Mars, first to be implemented with the Mars 2020 lander [38], and asteroids [60]. These activities have been addressed by various projects involving several NASA centers, universities and industry.

The main capabilities in the research focus are terrain relative navigation for precise landing, Hazard Detection and Avoidance (HDA), and Hazard Relative Navigation (HRN) for safe landing. In the wake of this research several algorithms and sensors have been developed. These shall be briefly discussed here.

When following the proposed lunar landing profile in [36] the first new-generation instrument to be used will be the long range altimeter [69], currently developed at NASA's Langley Research Center. It is designed to work from a range of about 30 km down to landing. When operated continuously, the sensor's data can be processed into a height profile. The navigation approach is to match this profile with an on-board Digital Elevation Model (DEM) for determination of the current position given in the Moon-Centered Moon-Fixed (MCMF) frame [39]. This function requires an a priori knowledge of the position of about 1.6 km. This instrument has been tested on the Vertical Testbed (VTB) Morpheus in 2012 and 2014.

Another approach for terrain relative navigation is pursued at NASA's Jet Propulsion Laboratory (JPL) [38]. The team uses an image taken by a monocular camera and then match it to an on-board map. The method works as follows. Based on the current navigation solution the camera image is rectified to agree with the on-board map's projection type. Since the on-board map covers the complete landing ellipse (for Mars landing) or parts of the descent orbit and powered descent (for lunar landing), it is too big for performing a matching operation. Therefore, by using the current navigation solution, a sub-map is cropped out from a low resolution version of the on-board map. Then the warped camera image is matched with the map in a coarse step. With the matching result, a second sub-map can be cropped from the full resolution on-board map; this time smaller in its geographical coverage but in full resolution. Also, instead of using the complete camera image, up to 100 small regions are extracted and correlated with the map. By including data from a co-registered DEM, the 3D position of each extracted region is ascertained. Over several images the 3D positions of the features and the Inertial Measurement Unit (IMU) measurements are fused, finally converging into a pose solution. Similar to the first Terrain Relative Navigation (TRN) method, this one requires a priori state knowledge in the order of 3 km for the position and 0.15 deg for the attitude. The system was tested on a helicopter flight in 2014 and as part of the Autonomous Descent and Ascent Powered-Flight Testbed (ADAPT) payload on Masten Xombie in 2014 [87].

The two sensors mentioned above provide estimations of the absolute position. The third instrument is a Doppler LIDAR. It provides terrain relative velocity and altitude in three directions simultaneously by using three sensor heads [2]. It is currently planned to use this sensor in the final part

of the trajectory, beginning at 4000 m altitude. Generation 2 has been tested on Morpheus in 2014, the third generation is to be tested as part of the CoOperative Blending of Autonomous Landing Technologies (COBALT) experiment on the VTB Xodiac in 2017 [12].

A very promising approach for landing site evaluation is to directly measure the topography of the landing site and to analyze the resulting DEM for safety figures such as roughness or slope. NASA foresees a Flash Light Detection and Ranging (LIDAR) instrument for this application. The current generation measures the underlying terrain at a maximum distance of 1.3 km and with a 128 x 128 pixel resolution at 20 Hz [72]. This system has been tested on Morpheus in 2014. However it has been found that the technology needs to be further advanced in terms of resolution, range precision, and data post-processing [3].

For several reasons, such as accumulated navigation dispersion, the current position of the spacecraft may diverge too much from the reference trajectory. Also the initially intended landing site may change due to a re-targeting as a consequence of landing site evaluation. In both cases the determination of a new reference trajectory may be necessary or even mandatory for ensuring mission success. Furthermore the necessary calculations will have to be performed on-board for ensuring an autonomous operation. Such a technology is the Guidance algorithm for Fuel Optimal Large Diverts (G-FOLD), to be applied for landing on the Moon and Mars. It has been tested as part of the ADAPT payload on flights on Xombie in 2014 [87].

In summary, the current state of NASA's approach for a safe and precise landing is to use state-of-the-art sensors such as an IMU and a Star Tracker (STR) to augment them with measurements from a long range altimeter and a camera-based terrain matching during the descent orbit and powered descent. During the late phase of the powered descent a Doppler LIDAR takes over from the long range altimeter for providing slant range and high precision velocity. These sensors should ensure an arrival of the spacecraft within 90 m of the intended landing site. Once arrived it is planned to measure the landing site in 3D with a Flash LIDAR. After selection of a safe landing site, the Flash LIDAR's continuing measurements shall be used to navigate relative to the selected landing site and finally to land with a precision of 3 m relative to it. The necessary adaptation of the flight path is to be performed by the G-FOLD technology.

Recently the Doppler LIDAR, the ADAPT technology comprising the camera-based terrain relative navigation, and G-FOLD have been integrated into the COBALT payload to be tested in 2017 [12]. With regard to the Flash LIDAR, the current focus is on its technology advancement [3]. This explains why no apparent future activities towards further flight testing could be identified. However it is expected that NASA is going to integrate a then-improved Flash LIDAR with the COBALT technology.

European Space Agency (ESA)

In recent years usually one major project was and still is setting the pace for the ESA activities in the area of autonomous, precise and safe landing. In 2007 after various studies ESA announced the project Next Exploration Science and Technology Mission (NEXT) lunar lander with the requirement of demonstrating high precision landing with hazard avoidance [83]. The project was later renamed to Lunar Lander and reached phase B1 before it was canceled during the Ministerial Council in 2012. However for some

aspects of the lunar lander the activities continued, including parts of the GNC system. Currently ESA is collaborating with the Russian Space Agency to supply the Precise Intelligent Landing using On-Board Technology (PILOT) system to be used in future Russian Moon landers. It is being designed to provide all necessary information to perform a precise landing and also hazard avoidance [34].

Several technologies are expected to help to achieve that goal. In the following these are explained in more detail: first the camera-based navigation techniques and then the hazard detection and avoidance techniques. One or more monocular cameras will be used for providing images which are then processed for relative and absolute navigation information.

The absolute navigation method provides position and attitude in the MCMF frame. The approach is to match craters detected in a camera image with an on-board database. The database is generated before the flight by extracting craters from geo-referenced DEMs and images. The matching is performed in 2D image coordinates. To do so the craters from the database are transformed into the image space by using the current pose knowledge from the navigation filter. Due to the detection in the image, the detected craters are of course already known in image space. Both point clouds are then matched by a nearest neighbor method. The absolute navigation method requires an a priori known Moon-relative pose. The required attitude precision is given with 0.1 deg [58]. It is planned to use this method starting approximately ten minutes before the Powered Descent Initiate (PDI) – the begin of the powered descent.

The relative navigation part is intended to stabilize the IMU-based propagation of position and attitude. The approach is to detect the optical flow between two consecutive images and to pass this information to the navigation filter, where it is fused to reduce the drift. The optical flow is determined by using a variant of the Kanade-Tomasi corner detector [24].

Two types of sensors are foreseen to provide the measurement data for hazard detection and avoidance. A scanning LIDAR is intended to be used for measuring the 3D structure of the landing site. From the measurements a DEM will be determined, establishing the necessary input for extracting hazard figures such as slope and roughness. Additionally camera images shall be used for determining further hazard maps such as shadows [17].

The first implementation of PILOT shall be a demonstration hardware to be flown on the Luna-25 or Lunar Glob mission [34]. There, a monocular camera will be demonstrated. Its main task is to capture images for later ground processing. A launch is planned for no earlier than 2019. The second implementation of PILOT shall be a fully operational system comprising one or more monocular cameras for relative and absolute navigation and a scanning LIDAR for hazard detection. A reliable launch date could not be found.

China

In December 2013 China successfully performed an autonomous and safe landing with the Chang'e 3 lander. It is the only lander so far which landed not only softly, but also performed a hazard detection and avoidance. This is of course a remarkable achievement. The mission profile was similar to the approaches of planned Moon landings of other agencies. It comprised a lunar circular parking orbit, a descent orbit, a powered descent, a final approach, and certain intermediate phases with the purpose of blending the states between the phases. The navigation system comprised an IMU,

an STR, sun sensors, a microwave altimeter and velocimeter, a Laser Altimeter (LA), a landing camera and a scanning LIDAR [50, 93].

Even though we would expect some kind of absolute navigation scheme to be applied during the descent orbit or the powered descent, we were not able to find any indications in literature. According to [50], navigation of the lander relied on inertial sensors, after receiving a last ground update ten minutes before PDI, which started at 15 km altitude. Below an altitude of 8 km the laser-based and microwave-based altimeters were used to update the propagation. Below an altitude of 4 km also measurements of the microwave velocimeter were included.

The hazard detection and avoidance system of Chang'e 3 comprised a two-stage approach. At first images were taken by the landing camera at an altitude of 2400 m and then processed on-board. This step is called coarse hazard detection and identified objects bigger than 1 m. Based on this analysis a preliminary landing area was selected. The second step was initiated at an altitude of 100 m above the landing area. There a scanning LIDAR [93] was used to generate a DEM of a 50 m × 50 m section of the underlying terrain. Based on the DEM two parameters were determined. One was the 10 m baseline slope and the other was the roughness showing hazards bigger than 20 cm. Using these parameters a landing site was finally selected.

It can be presumed that a safe landing site is surrounded by dangerous terrain. Therefore there is little margin for landing errors with respect to the selected landing spot. Chang'e 3 landed approximately 1.5 m from the selected site.

Background Summary
Optical navigation sensors augment state-of-the-art navigation systems with low-latency measurements mapping onto essential parts of the state such as terrain relative altitude, velocity and attitude, and possibly also attitude and position with respect to the target body's reference frame. Due to these compelling characteristics optical navigation will be part of any future autonomous landing system which is intended not only to land autonomously but also precisely and safely. Since the beginning of the ATON project in 2010, this argument remained and continues to be a compelling one and – in the hindsight of successful demonstrations (see below and in Chap. 4) – has gained even more power ever since.

1.4 Assumptions and Decisions

ATON targets the navigation system development for a landing on solar system bodies in general. The navigation system shall use the surface (respectively terrain) of the target body for obtaining the navigation solution (position and attitude in a target body-fixed coordinate frame). Although there is a high diversity in the size and structure of solar system bodies, it is only the atmosphere that has a high impact on the navigation system architecture. Based on this, the class of potential targets was narrowed down for ATON to celestial bodies with no or very thin atmosphere. This allows observing features and landmarks on the ground already from high altitudes. The selected class of targets includes the Moon, asteroids, comets, and other small planetary moons like Phobos and Deimos. Out of this class the Moon was selected as the reference target. The Moon is one of the

largest bodies without a significant atmosphere. Since the dynamics of a descent and landing are driven by gravity, the landing in a high gravity environment requires a higher accuracy and a faster control loop. Thus we can expect for a landing on the Moon challenging requirements for the GNC subsystem. A second aspect for choosing the Moon is that it is well known and well mapped with a lot of data publicly available.

When neglecting cooperative targets such as landing sites equipped with beacons [85], currently optical navigation techniques based on image processing can provide absolute and relative navigation information within the local reference frame of the target celestial body. Thus the work in ATON assumes that the following sensor suite is available for implementation in a future exploration mission using ATON's technology:

- Inertial Measurement Unit (IMU) providing measurements of the angular rate and the non-gravitational acceleration,
- Star Tracker (STR) providing inertial attitude information,
- Laser Altimeter (LA) delivering the distance to the ground along its line of sight,
- Monocular monochrome navigation camera taking images of the target body and terrain which are subject to further image processing, and
- Flash LIDAR providing 3D-images.

This assumption is based on the review and analysis of past and currently developed missions and technologies as well as on preliminary analyses at the beginning of the project. More details will be provided in Chap. 2.

2 System Overview

In order to define a goal for the technical development within the project ATON this chapter introduces the requirements for the targeted optical navigation system. For that purpose first a review of historic and planned missions (at the time of the start of the project in 2011) is undertaken. From that the targeted navigation accuracy is derived. As a next step a reference mission for the project is defined including the sensor suite and assumed characteristic performances.

2.1 Review of Historic and Planned Missions

Landing on planets and other celestial bodies has been and will be one of the fundamental prerequisites for the exploration of our solar system. It has been achieved in the past and many more future missions are planned. For establishing requirements and a reference mission these missions are analyzed.

2.1.1 Navigation Accuracy

For the review the historic NASA missions Surveyor and Apollo have been studied. Table 2.1 lists the GNC performance of these historic missions. The main goal of the Surveyor program was to validate a soft lunar landing. Although the landing sites had been defined, no specific requirements on the precision were given. During the Apollo missions the landing precision increased significantly. In the second Apollo mission pinpoint landing capability was to be demonstrated and achieved via landing the Lunar Module (LM) on purpose near Surveyor 3. Although both Surveyor and Apollo are considered as soft landing missions, the requirements demanded lower touch-down velocities for Apollo. This could be achieved with touch-down velocities in the order of 1 m/s.

Table 2.1: Results of historic soft lunar landing missions

Mission	Landing Precision	Touch-Down Conditions
Surveyor 1	19 km off target [71]	$v_v = 3 - 4$ m/s $v_h = 0.5$ m/s [64]
Surveyor 3	3 km off target [71]	$v_v = 1 - 2$ m/s $v_h = 0.3 - 0.9$ m/s [64]
Apollo 11	5.5 km off target [63]	$v_v = 0.5$ m/s $v_h = 0.5$ m/s [61]
Apollo 12	0.2 km off target [62, 63]	$v_v = 1$ m/s $v_h = 0$ m/s [62]

As future planned missions the NEXT lunar lander of ESA and NASA's planned Altair lander have been studied. The navigation-related requirements for these missions are shown in Table 2.2. Compared to those of the historic missions it can be seen that their accuracy has been increased significantly. One of the main changes is the higher demand for precision of the touch-down point. Additionally all maneuvers have to be performed autonomously after PDI which also includes the selection of a safe landing site.

2.1.2 Mission Profile

Besides the navigation accuracy the profiles of the missions, the used sensors and actuators have been analyzed. Following the review a typical mission profile comprises several major phases:

- Park orbit – position in a circular orbit of ≈100 km altitude
- Descent orbit – Descent Orbit Injection (DOI) into a Hohmann transfer orbit of 100 km × 10 km altitude
- Powered descent – PDI at 10 km, reducing most of moon-relative speed
- Approach – pitch over of vehicle at altitude of 1 km to 2 km, controlled descent rate, landing site evaluation, potentially retargeting
- Terminal descent – 30 m altitude over landing site, horizontal speed removed, descent until touch-down

2.1.3 Derived Top-Level Requirements for Future Lunar Pinpoint Landing Missions

Based on the analysis of the reviewed missions a set of top-level requirements can be formulated as:

General requirements:

- The lander shall be able to perform an autonomous, precise and safe landing on a pre-defined landing site.
- The lander shall provide global access; the whole lunar surface shall be selectable as landing site.

Touch-down conditions

- The landing shall occur with a precision of 200 m 3-σ at a pre-defined landing site on the lunar surface.
- The lander shall be able to autonomously select a safe landing site within the precision-radius.

Table 2.2: Requirements of future lunar soft landing missions

Mission	Landing Precision	Safe Landing Site Conditions	Touch-Down Conditions
Altair [16]	30 m 1-σ global	local slopes <5 deg stones or holes <30 cm	$v_v < 2$ m/s $v_h < 1$ m/s [14]
NEXT [33]	200 m 3-σ south pole	local slopes <15 deg surface roughness <50 cm select safe site with 99 % probability	TBD (order of 1 m/s)

- The lander shall be able to land within the precision radius of the selected landing site.
- The touch-down speed shall not exceed 1 m/s in vertical velocity and horizontal velocity.

This set will form the basis for a more detailed requirements definition.

2.2 ATON Development Goals

In this section the requirements for the navigation system to be developed within the project ATON will be defined. At first the navigation accuracy requirements from literature are discussed. Then the navigation accuracy requirements are defined.

2.2.1 Initial Navigation Accuracy Requirement Definition

The top-level navigation requirements from Sect. 2.1.3 provide an overview of the desired landing precision and landing conditions of a safe and precise lunar landing. To achieve these goals, the navigation system must support the GNC system with sufficient state-vector data. In the case of ATON the state-vector shall be autonomously determined from the beginning of the landing maneuver at the DOI. In the following an initial definition of the required navigation precision during all landing stages shall be found. The definition shall serve as a guideline for the first development steps.

To support the definition process, data from a covariance analysis of a lunar landing navigation system [22] are taken as reference. The goal of the analysis was to find the requirements for a Terrain Relative Navigation (TRN) sensor to achieve a 100 m (3-σ) navigation accuracy at landing. Also the landing accuracy is studied. For that purpose the navigation data is fed into a proportional derivative controller which controls position and attitude. The study's focus was directed to the navigation performance. The authors point out that the controller might not have evolved to a fully optimized state, especially in terms of dispersion control. Hence, the numbers are considered to be a first baseline for the required navigation performance. The TRN sensor is accompanied by a sensor suite shown in Table 2.3.

Table 2.3: Navigation Instruments of lunar landing covariance study [22]

Instrument	Measurement
IMU	3-axis acceleration and angular rate
STR	inertial attitude
Altimeter	altitude
Velocimeter	3-axis velocity measurement
TRN sensor	position in landing site coordinate frame during PD

The mission follows a profile consistent with the ones surveyed in Sect. 2.1.2. It performs a DOI to initiate a Hohmann transfer, which ends at PDI after half a lunar orbit. The Powered Descent (PD) is finished at the approach maneuver, where most of the horizontal velocity is removed and

a nearly vertical descent towards the landing site begins. The autonomous navigation is initiated by starting the propagation in the park orbit. Before the DOI a last external update of the state-vector is determined by tracking data of the Deep Space Network (DSN). During Descent Orbit (DO) the propagation only uses STR and IMU gyro data. The accelerometers of the IMU are used only during thruster action. Several minutes in advance of the PDI the altimeter begins measurements. The TRN sensor is working between 2 and 8 km altitude and performing position measurements every 10 s. At an altitude of 2 km also the velocimeter starts measuring.

A major outcome of the study is the determination of the main errors of propagation. One error is represented by the accuracy of the initial state-vector. For a typical Hohmann transfer orbit from DOI to PDI, 1 m of initial navigation error at DOI adds roughly 4 m of downrange navigation at PDI. The study assumes an initial (3-σ) state-vector error of 1500 m downrange, 200 m cross-track and 50 m in altitude, which has been mentioned as being consistent with the ALHAT program. The second major source for propagation errors is the quality of the gravity model. An error of 10 mGal (1 Gal = 0.01 m/s^2) causes roughly an error of 2 km downrange after half a lunar orbit of coasting in DO. The study assumes a gravity model error of 20 mGal.

For a better understanding of the following discussion and in Sect. 2.2.2, a short overview on dispersion control is given in this paragraph. Any navigation error leads to a dispersion of the lander's position from the reference path. With a high probability, the dispersion is at least in the same range as the navigation error. Nominally the spacecraft (SC) is designed for an optimized reference path, which represents the most fuel efficient way to land on the Moon. The earlier the dispersion can be measured, the more efficient it can be controlled by slight changes to the reference path. The later the dispersion is measured the higher the modifications of the remaining reference path have to become. This leads to increased fuel consumption. The main part of landing dispersion is mainly in the down-range direction. An excellent possibility for down-range dispersion control is at the PDI. By changing the time of thruster ignition by several seconds and by slight modifications to the PD reference path the down-range dispersion can be reduced down to the navigation error with very little fuel cost.

In Table 2.4, the required navigation accuracies for position and velocity for a 100 m (3-σ) landing navigation accuracy are displayed. There was no given data for the speed accuracy during PD, approach and landing.

The initial navigation accuracy at DOI reflects the necessary precision of the last tracking update. During the orbit the position and velocity determination accuracy degrades due to a propagation which is only supported by IMU and STR measurements. The highest error of 10 km occurs at the point just before the altimeter starts working. This is roughly one or two minutes before the PDI. After this moment the accuracy continuously increases due to downrange-altitude correlations. At PDI the navigation error amounts to 5000 m. Thus the dispersion can be reduced from 10 km to 5 km by modifications of the thruster ignition time of ≈3 s. At the altitude of 8000 m the TRN sensor starts working and updates the propagation with lunar surface relative position measurements. The TRN sensor stops at 2000 m where the navigation accuracy has improved to a value of less than 100 m. Thus, during the last three minutes of the landing maneuver the remaining dispersion of at least 5 km has to be corrected. Due to missing TRN measurements during the vertical descent the propagation errors add up to a 100 m navigation error at the time of landing.

Table 2.4: Required navigation accuracy for a 100 m (3-σ) lunar landing from [22]

Mission Phase	Autonomous Position Determination (3-σ)	Autonomous Velocity Determination (3-σ)
DOI[1]	dr: 1500 m cr: 200 m alt: 50 m	dr: 0.047 m/s cr: 0.2 m/s alt: 1.5 m/s
pre PDI[2]	mainly dr: 10 000 m	no data
PDI	mainly dr: 5000 m	no data
PD before TRN	mainly dr and cr: 2000 m	no data
PD after TRN	mainly dr and cr: 70 m to 80 m	no data
landing	mainly dr and cr: 100 m	no data

[1] accuracy of DSN tracking

[2] beginning of altimeter function

Considering the analysis in [22] a potential TRN sensor has to have a position measurement accuracy of ≈40 m to 80 m (3-σ).

2.2.2 ATON Navigation Accuracy Requirements Definition

The analysis carried out above shows that an autonomous and precise navigation for a lunar landing is feasible. The navigation was performed by propagating the orbit with support of inertial measurements, altimeter, velocimeter and a TRN sensor. The main reasons for propagation errors turned out to be the initial navigation solution and the quality of the gravity model. From the moment the navigation solution was updated with precise TRN and altimeter measurements the propagation proved to be stable with low frequency updates of the TRN sensor. The necessary TRN sensor precision is sufficient to be within the range of the required landing precision. These findings shall serve as the basis for the definition of the navigation requirements for ATON.

For ATON the following assumptions are made:

- The IMU and STR used in ATON are of comparable quality to the instruments mentioned in Table 2.3,
- The initial state-vector precision is comparable to [22],
- Absolute position measurements are available in parts of the DO,
- No altimeter function is available before PDI (only via optical position measurements),
- During PD, altimeter and velocimeter function is performed by the optical navigation system,
- The 3D imaging system is working after the landing site becomes visible.

Based on these assumptions the required navigation performance for ATON is shown in Table 2.5.

At the DOI the navigation accuracy corresponds to the capability of the ground station network. During the coasting in the DO the landmark navigation system provides several measurements with an accuracy of 1 % of current height or 100 m to 1000 m for down-range and cross-range and 0.5 % of current height or 50 m to 500 m for altitude. This enables the

Table 2.5: Required navigation accuracy for a 200 m (3-σ) lunar landing for ATON

Mission Phase	Autonomous Position Determination (3-σ)	Autonomous Velocity Determination (3-σ)
DOI[1]	dr: 1500 m cr: 200 m alt: 50 m	dr: 0.047 m/s cr: 0.2 m/s alt: 1.5 m/s
pre PDI	mainly dr: 100 m to 1000 m	0.5 m/s
PDI	mainly dr: 100 m	0.5 m/s
approach: before 3D imaging	mainly dr: 500 m	0.5 m/s
approach: after 3D imaging	mainly dr and cr: 50 m	0.5 m/s
landing	2 m	0.1 m/s

[1] accuracy of ground station tracking

propagator to determine the SC position at PDI within 100 m. An early dispersion control can be performed at PDI via igniting the main engines earlier or later with respect to the reference PDI. Because there is no dispersion control during coasting in DO (no thruster action planned in DO), the dispersion can grow to an error of 10 km. This value is comparable to the navigation error before PDI in Table 2.4. Due to the relative inexperience of the European space community in navigation in the lunar gravity field this error might even grow. On the other hand the outcome of missions like Kaguya or the mission Gravity Recovery and Interior Laboratory (GRAIL) [25] might significantly reduce the gravity field error. Thus a 10 km dispersion will be assumed. This can be counteracted by a 6 s advanced/delayed ignition of the main engine.

During PD the optical navigation system will perform altimeter and velocimeter functions. Due to the lack of position measurements the navigation error will grow during this period. The task of the navigation system is to keep the propagation stable and the error growth small. In Sect. 2.2.1 this is shown to be feasible. This error shall not be greater than 500 m.

After the pitch over and initial coming visibility of the landing site, the 3D imaging system will start to take measurements. The resulting data will possess an initial resolution in the order of 50 m and will continuously grow during the descent. The 3D data will be compared with an on-board 3D map of the landing site, achieving a navigation knowledge in the order of 50 m. The purpose of the 3D imaging system is also to deliver the necessary data for the evaluation of the landing area. When a safe landing site is found, the GNC system must be able to place the lander inside the safe area. The size of the safe area is assumed to be in the order of three times the diameter of the lander. Thus, the allowed landing error is in the order of half a lander diameter. The navigation requirement for the landing is therefore set to 2 m. This should be possible, when considering the 3D data requirement at the final stage of the landing. The required 3D resolution is in the order of 15 cm per pixel. This data will become available at an altitude of ≈400 m.

2.3 Reference Mission

In order to support further detailed development a reference mission has to be designed. Following the assumptions and decisions in Sect. 1.4 the baseline is a lunar landing.

2.3.1 Sequence of Events

For generating simulation data the reference mission for a lunar landing has to be defined based on the analysis results from Sects. 2.1.2 and 2.2.2. The general sequence of approach and landing is defined as:

1. Start in an initial 100 km × 100 km quasi circular orbit around the Moon
2. Execution of Descent Orbit Injection (DOI) maneuver to reach a 100 km × 10 km orbit
3. Flight along the elliptic descent orbit to pericenter
4. Start of powered descent (PDI) close to the pericenter of the descent orbit
5. Achievement an almost vertical descent for the last 100 s
6. Final conditions: altitude ≈1 m above landing site at <1 m/s velocity

Figure 2.1 illustrates the sequence of events for the reference mission.

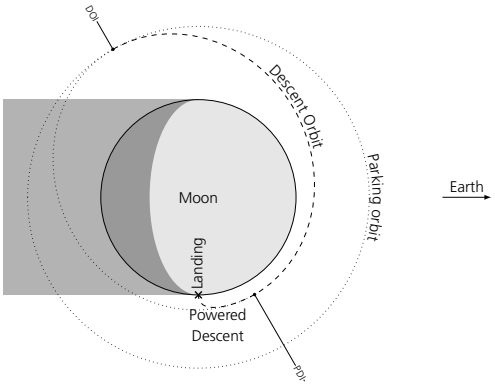


Figure 2.1: Sketch of the lunar landing scenario with a polar orbit

2.3.2 Landing Site and Date

Several promising sites have been selected as a potential landing site. Furthermore, four arbitrarily chosen sites have been selected. For some of them a landing at different times has been considered in order to see the effect of different illumination conditions. Table 2.6 shows the list of scenarios considered in the project. For a systematic analysis the scenarios 6 to 14 have been used since they offer on one hand three sites with different topographic environments. On the other hand with these scenarios three different landing times and therefore the effect of different illumination conditions can be analyzed.

2.3.3 Powered Descent Trajectory

A powered descent trajectory with constraints for actuators and flight states as well as the objective of minimal fuel consumption can only be generated as a solution of an optimal control problem. For the specific case of a landing vehicle with non-throttable engines a solution is provided in [65]. This paper defines an optimal control problem and provides a solution. Furthermore a tracking controller is designed which enables the vehicle to follow the designed reference trajectory even in the presence of uncertainties. A more robust implementation of an on-board algorithm is presented in [52] where a suboptimal trajectory is interpolated on-board depending on the initial state. This allows very large uncertainties under the initial conditions at PDI.

Fig. 2.2 shows the altitude vs. downrange profile of the powered descent. Fig. 2.3 displays the velocity profiles. It can be seen that the main thrust is

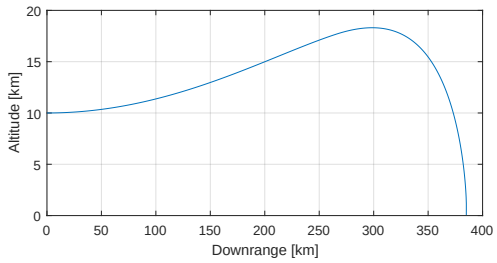


Figure 2.2: Altitude vs. Downrange of the powered descent trajectory as computed in [65]

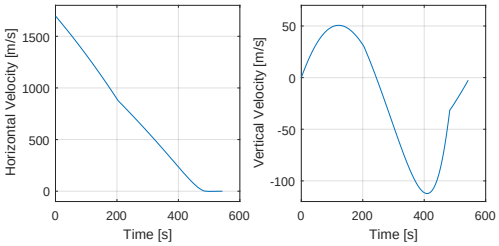


Figure 2.3: Horizontal and vertical velocity vs. time for the powered descent trajectory as computed in [65]

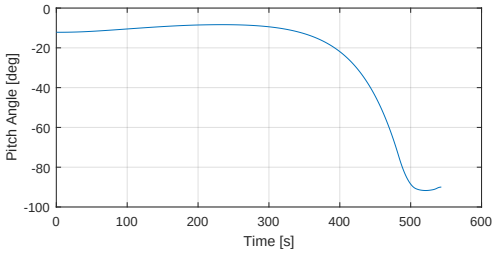


Figure 2.4: Pitch angle vs. time for the powered descent trajectory as computed in [65]

changed in only three steps. This meets the specific requirements of a landing with non-throttable engines where the thrust is reduced by switching of pairs of engines.

In Fig. 2.4 the pitch angle is shown for the powered descent. It can be seen that the pitch angle is kept at low angles (between 0 and 20 deg) for a long period where mainly the horizontal velocity is decreased. Then, the landing vehicle pitches down so that the x-axis (down direction when the lander is landed) points more and more down. In order to provide good visibility of the landing site for the on-board sensors the last part of the descent is almost vertical with a pitch angle close to –90 deg.

Before PDI the spacecraft follows an elliptical descent orbit from an altitude of 100 km. For the project ATON a period of 2600 s before PDI is included in the scenario in order to provide sufficient time for the acquisition phase of the navigation system in order to achieve the navigation accuracy specified in Sect. 2.2.2.

Table 2.6: Landing scenarios analyzed in ATON

#	Name	Latitude [deg]	Longitude [deg]	Date (UTC)	Heading
1	Apollo 12 - RD	-03.0124	336.5874	21.01.2016, 01:00:00	215
2	Shackelton Rim 1	-89.7788	206.5651	03.06.2016, 01:00:00	180
3	Connecting Ridge	-89.4632	222.5104	01.06.2016, 01:00:00	180
4	Kimura Crater	-57.1000	118.4000	10.06.2016, 01:00:00	180
5	Test 1	0	0	19.01.2016, 01:00:00	215
6	Test 2	-5	8	18.01.2016, 20:00:00	180
7	Test 2a	-5	8	19.01.2016, 20:00:00	180
8	Test 2b	-5	8	19.01.2016, 08:00:00	180
9	Test 3	5	7	18.01.2016, 20:00:00	180
10	Test 3a	5	7	19.01.2016, 20:00:00	180
11	Test 3b	5	7	19.01.2016, 08:00:00	180
12	Test 4	-5	6	18.01.2016, 20:00:00	180
13	Test 4a	-5	6	19.01.2016, 20:00:00	180
14	Test 4b	-5	6	19.01.2016, 08:00:00	180

2.4 System Design

In addition to the definition of requirements and the definition of the mission a reference system design is established for the project ATON. As pointed out earlier the goal of the project is to develop a generic system and technologies for optical navigation which should be applicable to various space exploration missions. This section defines a reference set of sensors to be included in the analysis. For the selected reference mission their parameters are defined. Finally, a system architecture is designed for the reference mission based on selected navigation and image processing technologies to be developed during the project ATON.

2.4.1 Reference Sensor Configuration

Based on the analyses in Sect. 2.1 the following sensors are included in the navigation system:

- IMU providing measurements of the angular rate and the non-gravitational acceleration,
- STR providing inertial attitude information,
- LA delivering the distance to the ground along its line of sight,
- Monocular monochrome navigation camera taking images of the target body and terrain which are subject to further image processing, and
- Flash LIDAR providing 3D-images.

For the simulation of input data for the navigation and image processing algorithms and methods the parameters of the sensors must be fixed. These parameters are a baseline for further development steps. For some later analyses some parameters can be changed too.

For the IMU and the STR three different classes of sensors are defined. This allows for the analysis of the impact of sensor accuracy on the navigation accuracy.

2.4.1.1 Star Tracker (**STR**)

Table 2.7 lists the parameters for the three different accuracy classes of star trackers used in ATON.

For the alignment of the star tracker the following conditions are considered. It shall point away from the Sun and lunar surface. The baffle exclusion angles listed in Table 2.7 must be met at all times. Furthermore, the plume of the main engine shall not be included in the Field of View (FOV) of the sensor. During the landing the vehicle performs a pitch of about 90 deg where the baffle exclusion angles also have to be considered.

Since most landings on the Moon occur on a lunar morning (in order to have about 14 days of illumination) it can be assumed that the Sun elevation at the landing site is not very high. Furthermore, it can be assumed that the low elevation of the Sun is not in flight direction or anti-flight direction since in this case either the navigation camera may be blinded by the Sun or would not see shadows and therefore would have only very few characteristic features for optical navigation. Based on these assumptions the Sun elevation will be below 60 deg and the Sun azimuth with

respect to flight direction is between 30 deg and 150 deg or –30 deg and –150 deg respectively.

With these conditions the STR should be mounted with the following transformation

$$\boldsymbol{R}^{b}_{\mathrm{STR}} = \begin{bmatrix} \cos\alpha_{y,\mathrm{STR}} & 0 & \sin\alpha_{y,\mathrm{STR}} \\ 0 & 1 & 0 \\ -\sin\alpha_{y,\mathrm{STR}} & 0 & \cos\alpha_{y,\mathrm{STR}} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha_{x,\mathrm{STR}} & \sin\alpha_{x,\mathrm{STR}} \\ 0 & -\sin\alpha_{x,\mathrm{STR}} & \cos\alpha_{x,\mathrm{STR}} \end{bmatrix} \tag{2.1}$$

where the angles to rotate the STR frame to the body-fixed frame are defined as

$$\alpha_{x,\mathrm{STR}} = \delta \cdot 40\,\mathrm{deg} \tag{2.2}$$

$$\alpha_{y,\mathrm{STR}} = 180\,\mathrm{deg} + 40\,\mathrm{deg} \tag{2.3}$$

$$\delta = \begin{matrix} 1 & \leftarrow \boldsymbol{e}^{b}_{\mathrm{Sun}} \cdot \boldsymbol{e}^{b}_{y} < 0 \\ -1 & \leftarrow \boldsymbol{e}^{b}_{\mathrm{Sun}} \cdot \boldsymbol{e}^{b}_{y} > 0\,. \end{matrix} \tag{2.4}$$

With this definition the z-axis of the STR is the boresight. The quantity δ is -1 or 1 if the Sun is in the right or respectively left hemisphere with respect to the flight direction. The angle $\alpha_{x,\mathrm{STR}}$ is chosen such that for a Sun elevation of 60 deg, the Sun exclusion angle is 70 deg and the exclusion angle to the horizon is 50 deg. The angle $\alpha_{y,\mathrm{STR}}$ is the sum of two angles. First, 180 deg are needed to point the STR to zenith since the z-axis of the lander points to nadir during horizontal flight of powered descent. Secondly, a rotation of another 40 deg is needed to point the STR away from the lunar horizon when the vehicle is landing vertically.

Table 2.7: STR specifications as used in the project ATON

Star Tracker Parameters	**Very Good**	**Good**	**Acceptable**
Accuracy [arcsec]			
Pitch - Roll axes	<1	1 to 10	10 to 60
Yaw Axis	<2	2 to 20	20 to 120
Slew Rate [deg/s]			
full performance	2	1	0.2
reduced performance	5	2	1
Attitude Acquisition Time [s] (lost in space)			
Nominal case	1	2	more than 2
Worst case	2	3	more than 3
Success Rate [%]	100	99	97
Update Rate [Hz]	10	5	1
FOV [deg]	25	15	10 or less
Baffle Exclusion angle [deg]			
Sun	30 or less	30 to 40	more than 40
Earth	20 or less	20 to 30	more than 30

Table 2.8: IMU specifications as used in the project ATON

	MEMS	**FOG/Servo**	**RLG/Quartz**
Gyroscope			
Bias [deg/h]	10	1	0.005
Scale factor [ppm]	1500	300	10
Noise [deg/$\sqrt{\text{h}}$]	1.5	0.03	0.005
Accelerometer			
Bias [mg]	20	2	0.02
Scale factor [ppm]	2000	500	100
Noise [μg/$\sqrt{\text{h}}$]	50	15	15
Alignment error [mrad]	0.5	0.5	0.2

2.4.1.2 Inertial Measurement Unit (IMU)

Table 2.8 lists the parameters for three different IMU accuracy classes used in ATON.

For the alignment of the IMU no specific conditions exist. Therefore, it is assumed for simplicity that it is located in the origin of the body-fixed coordinate frame and the axes are aligned with the body-fixed axes. Therefore the transformation can be described as

$$\boldsymbol{R}^{b}_{\text{IMU}} = \boldsymbol{I}_{3\times3} \tag{2.5}$$

$$\boldsymbol{r}_{b,\text{IMU}} = 0\,. \tag{2.6}$$

2.4.1.3 Navigation Camera (CAM)

For the navigation camera the following parameters have been selected based on the review of currently developed and planned missions and detailed geometric analysis. Table 2.9 shows the baseline parameters.

Table 2.9: Camera specifications as used in the project ATON

Resolution [px]	1024 × 1024
Frame rate [1/s]	30
FOV [deg]	40 × 40

The camera shall be aligned that it points during powered descent towards the lunar surface. During the final descent before landing the landings site shall be within the FOV. Furthermore, the main thrust direction (body-fixed x-axis) shall not be in the FOV to avoid disturbances from the main engine plume. In order to meet these conditions the camera is mounted with its boresight in the x-z-plane of the lander. The camera is rotated such that the border of the field of view is along the body-fixed x-axis.

The camera is mounted outside or on the surface of the landing vehicle. A diameter of about 4 m is assumed for the vehicle. This leads to a lever arm with respect to the IMU or body-fixed frame. Since LIDAR, LA and navigation camera are mounted on the same side of the lander they are separated along the body-fixed y-axis.

With these settings the transformation between camera frame and body-fixed frame is defined as

$$\boldsymbol{R}^{b}_{\mathrm{CAM}} = \begin{bmatrix} \cos\alpha_{\mathrm{CAM}} & 0 & \sin\alpha_{\mathrm{CAM}} \\ 0 & 1 & 0 \\ -\sin\alpha_{\mathrm{CAM}} & 0 & \cos\alpha_{\mathrm{CAM}} \end{bmatrix} \tag{2.7}$$

$$\boldsymbol{r}_{b,\mathrm{CAM}} = \begin{bmatrix} -0.5\,\mathrm{m} \\ 0.2\,\mathrm{m} \\ 2.0\,\mathrm{m} \end{bmatrix} \tag{2.8}$$

where the angle to rotate the CAM frame to the body-fixed frame is defined as

$$\alpha_{\mathrm{CAM}} = 90\,\mathrm{deg} - \frac{\mathrm{FOV}_{\mathrm{CAM}}}{2}\,. \tag{2.9}$$

The rotation of 90 deg is needed in order to define the z-axis of the camera as its boresight.

For further analysis the system concept will include a second camera. It will have the same parameters but a different viewing direction. Depending on the analysis this will be separately defined.

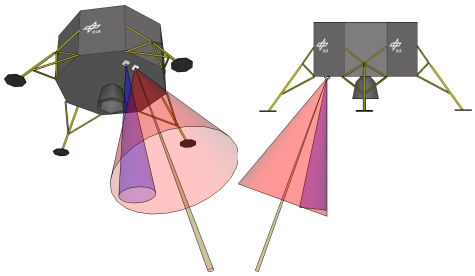


Figure 2.5: Sensor reference configuration: red - CAM, brown - LA, blue - LIDAR

2.4.1.4 Light Detection and Ranging (LIDAR)

For the LIDAR the parameters have been selected based on the review of currently developed and planned hardware [70]. Table 2.10 shows the baseline parameters.

Table 2.10: LIDAR specifications as used in the project ATON

Resolution [px]	400 × 400
Frame rate [1/s]	1
FOV [deg]	12 × 12
Range [m]	1 to 1000
Noise [m]	0.02

For the LIDAR sensor the same requirements as for the camera are applicable. With these settings the transformation between LIDAR frame and body-fixed frame is defined as

$$\boldsymbol{R}^{b}_{\mathrm{LIDAR}} = \begin{bmatrix} \cos\alpha_{\mathrm{LIDAR}} & 0 & \sin\alpha_{\mathrm{LIDAR}} \\ 0 & 1 & 0 \\ -\sin\alpha_{\mathrm{LIDAR}} & 0 & \cos\alpha_{\mathrm{LIDAR}} \end{bmatrix} \tag{2.10}$$

$$\boldsymbol{r}_{b,\mathrm{LIDAR}} = \begin{bmatrix} -0.5\,\mathrm{m} \\ -0.2\,\mathrm{m} \\ 2.0\,\mathrm{m} \end{bmatrix} \tag{2.11}$$

where the angle to rotate the LIDAR frame to the body-fixed frame is defined as

$$\alpha_{\mathrm{LIDAR}} = 90\,\mathrm{deg} - \frac{\mathrm{FOV}_{\mathrm{LIDAR}}}{2}\,. \tag{2.12}$$

The rotation of 90 deg is needed in order to define the z-axis of the LIDAR as its boresight.

2.4.1.5 Laser Altimeter (LA)

The altimeter shall measure the distance to the ground along a defined line of sight. Based on the data in [22] three different sensor types as listed in Table 2.11 have been used.

Table 2.11: LA specifications as used in the project ATON

Sensor Type	Range [km]	Noise [m]	Sample Rate [Hz]
1	20 -120	10	0.5
2	2 – 20	5	0.5
3	0.01 – 2	1	0.5

Apart from the direct altitude measurement the altimeter shall provide distance information supporting the image-based navigation using the camera. For creating a direct relation between image pixels and the distance measurement of the altimeter the line of sight of the altimeter shall lie within the FOV of the camera. For best use of this additional information the altimeter line of sight is aligned with the center of the camera image.

With the following transformation the z-axis of the altimeter is its line of sight:

$$\boldsymbol{R}_{\mathrm{LA}}^{b} = \begin{bmatrix} \cos\alpha_{\mathrm{LA}} & 0 & \sin\alpha_{\mathrm{LA}} \\ 0 & 1 & 0 \\ -\sin\alpha_{\mathrm{LA}} & 0 & \cos\alpha_{\mathrm{LA}} \end{bmatrix} \tag{2.13}$$

$$\boldsymbol{r}_{b,\mathrm{LA}} = \begin{bmatrix} -0.5\,\mathrm{m} \\ 0.4\,\mathrm{m} \\ 2.0\,\mathrm{m} \end{bmatrix} \tag{2.14}$$

2.4.2 Visibility Analysis

Before the defined sensor configuration is used in simulations and for the development of the optical navigation system a visibility analysis for the imaging sensors is carried out. For this analysis the trajectory from Sect. 2.3.3 is used. Figure 2.6 shows its geometry projected in the plane of the powered descent trajectory. The intersection of the horizon at the landing site with the trajectory marks the beginning of the visibility of the landing site for the landing vehicle. For the period from the begin of the visibility until touchdown the analysis is carried out.

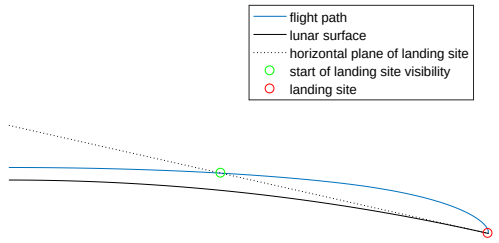


Figure 2.6: Geometry of powered descent trajectory

For the visibility analysis two angles are defined. First, the horizon look angle is defined as the angle between the line of sight to the horizon and the x-axis of the lander (down direction when vehicle has landed). Secondly, the target look angle or landing site look angle is defined as the angle between the line of sight to the landing site and the x-axis. Figure 2.7 explains the viewing angles.

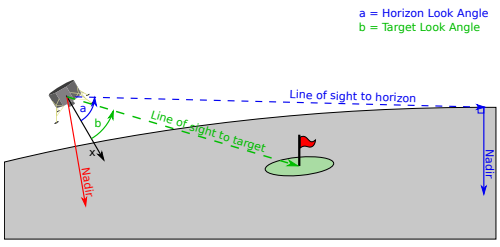


Figure 2.7: Definition of look angles for visibility analysis

The results of the analysis are shown in Fig. 2.8. The upper plot shows three angles vs. time:

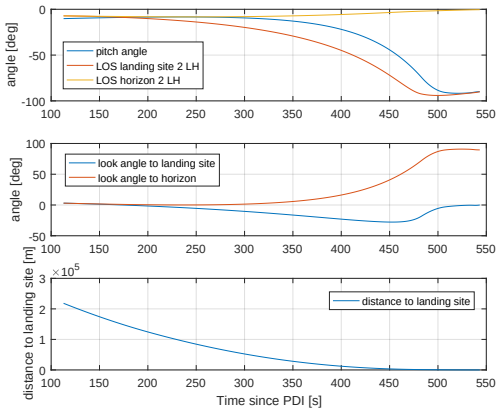


Figure 2.8: Visibility analysis: Evolution of look angles

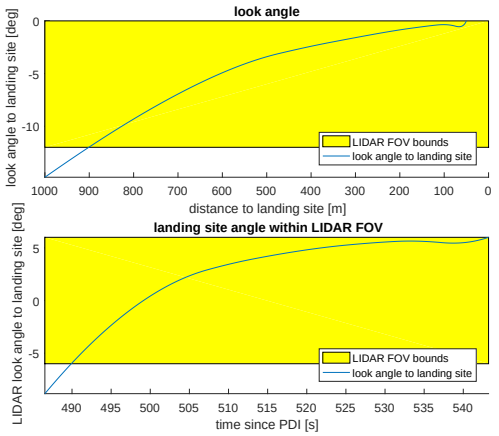


Figure 2.9: Visibility for LIDAR

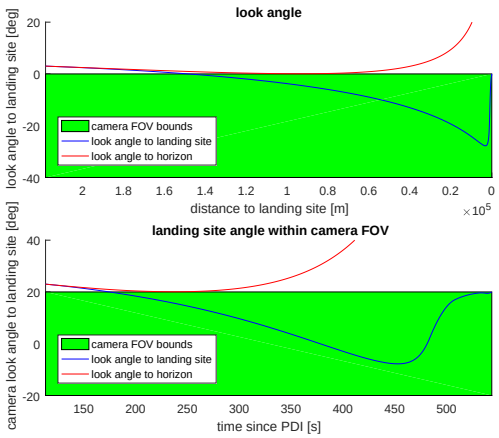


Figure 2.10: Visibility for navigation camera

1. The angle between the lines of sight to the horizon in flight direction and the local horizontal,
2. The angle between the line of sight to the landing site and the local horizontal, and
3. The pitch angle.

It can be seen that the pitch angle and angle to line of sight to the landing site follow a similar profile. The look angles as defined above are the differences of the line of sight angles to the pitch angle. This can be seen in the second plot. Since the profile of the pitch angle and angle to line of sight to the landing site are similar, only small values are reached for the look angle to the landing site. The third plot shows the distance to the landing site. It starts at about 200 km. Distances of 10 km and 1 km are reached at 409 s and 486 s, respectively.

The look angles are now compared to the selected FOV of both imaging sensors: the LIDAR and the CAM. Fig. 2.9 shows the look angle to the landing site and the landing site angle within the LIDAR FOV. In both plots the yellow area marks the FOV of the LIDAR. It can be seen that the landing site becomes visible in the LIDAR images at about 490 s which corresponds to a distance of approximately 490 m. This distance and time is sufficient for a landing site evaluation (hazard mapping) and for a 3D-based navigation during the final vertical descent.

Fig. 2.10 shows the look angle and the landing site angle within the CAM FOV. For the CAM the landing site appears in the FOV much earlier since it has a larger FOV. It starts at about 170 s which corresponds to a distance of about 150 km. From that point on the landing site is visible until touchdown. This allows a relative navigation to landmarks close to the landing site from the early phase of the powered descent.

The results of this visibility analysis show that the selected configurations provide a sufficient visibility for both imaging sensors during the powered descent.

2.4.3 System Architecture

In the sections outlined above, the set of sensors as well as their alignment on the landing vehicle have been defined. Since the output of the system shall be the navigation state vector, a mandatory element is a navigation filter which combines and fuses all sensor measurements and pre-processed data to a navigation solution. This is complemented with further modules for processing of image data. Fig. 2.11 shows the conceptual data flow within the ATON navigation system with seven processing modules including the navigation filter.

The processing modules are encapsulated in tasks, which are executed in parallel. The inter-module communication and the scheduling of the tasks is managed by DLR's data flow-oriented Tasking Framework [57]. It ensures that a module is only executed if all necessary inputs are available. The integration of the ATON software was conducted in a model-driven manner: an extended SysML/UML model was created to describe the processing modules with their interfaces and parameters, data types, priorities

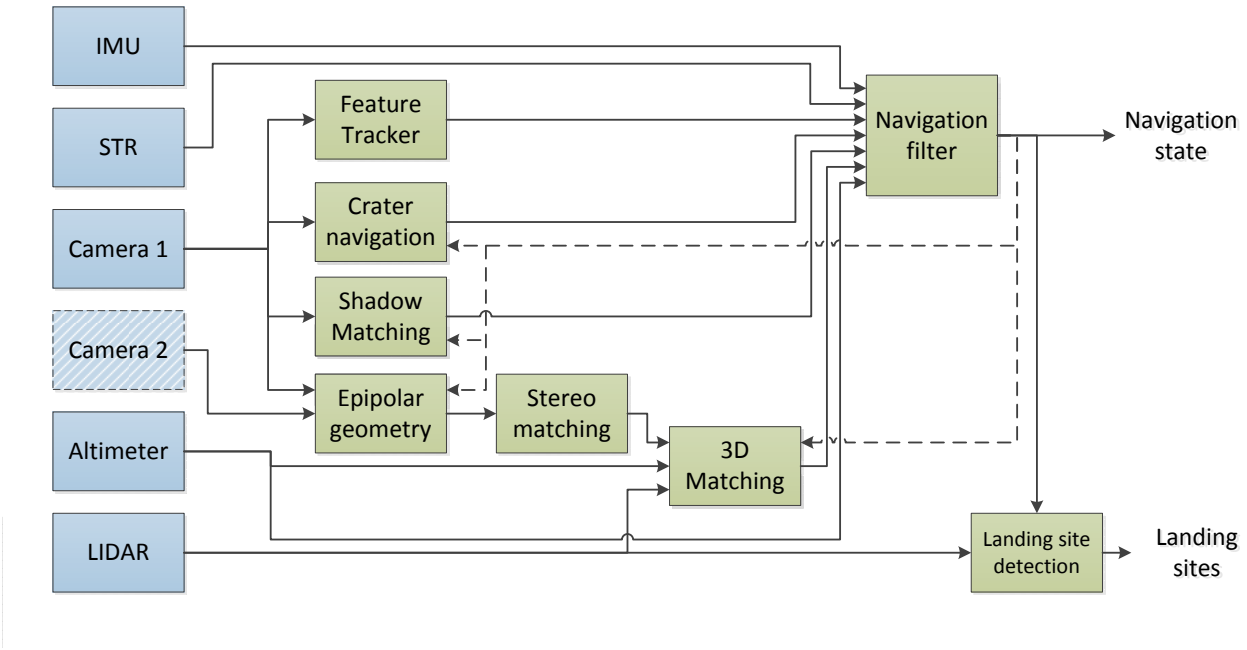


Figure 2.11: Block diagram of the ATON system

and the data flow between modules [19]. Custom code generators created the source code for data types, communication, module interfaces and serialization code for the telemetry.

All processing modules are described in more detail in Sect. 3. The following list provides a brief overview of each module.

Crater Navigation: The Crater Navigation module detects lunar surface impact craters in the images of the navigation camera, and by assigning each crater detection an element from a static crater catalog that is referenced in moon-fixed coordinates, a moon-fixed position can be computed. This position is supplied as a measurement to the navigation filter that may use it to cancel accumulated position and velocity error from the relative navigation. The crater detection is based on the extraction and matching of adjacent areas of above- and below-average brightness that model the reflection and shadow of typical crater interiors under illumination [55, 56].

Feature Tracker: The Feature Tracker module is used to extract and track image features in two subsequential input images. To perform this task the Lukas Tomasi Kanade (KLT) Tracker is used. This tracker is based on two steps: The first step is the image feature extraction based on high gradients in two axes [80], while the second step is the feature tracking which is based on image region similarity [53]. The 2D pixel coordinates of these image features in the two subsequent input images are sent to the Navigation Filter module.

Epipolar Geometry: The Stereo Matching, as used in the 3D processing chain, requires an accurate knowledge about the relative orientation of every two images being matched. It is required to calculate their epipolar geometry with less than 0.5 pixels of error to ensure the quality of the 3D model. The Epipolar Geometry module performs this task taking two subsequent images as input, together with the

rough relative orientation provided by the Navigation Filter. It extracts and matches common features between the two images and uses them to calculate the precise relative orientation between the two images using a small bundle adjustment with Random Sample Consensus (RANSAC). Finally, it passes the calculated relative orientation to the Stereo Matching module for each pair of images.

Stereo Matching: The Stereo Matching module computes dense depth maps from two consecutive and partly overlapping images, also known as structure from motion [82]. It uses the Semi-Global Matching (SGM) algorithm which is known from robotics and aerial image processing to provide accurate and dense depth maps [29–31]. Given two camera images with approximately 80 % overlap and the accurate relative orientation provided by the Epipolar Geometry module, the SGM can triangulate the so-called disparity for corresponding image points. As the ATON sensor suite contains calibrated cameras it is possible to convert the disparities into metric depth values. This allows for the provision of metric depth maps to the 3D Matching module.

3D Matching: The 3D Matching module provides an absolute pose estimation in the lunar reference frame. It is based on the Iterative Closest Point (ICP) algorithm [9, 13, 75] which can determine the offset, i.e. the relative transformation, between two 3D point clouds. The module can either use a range map from the Flash LIDAR or a metric depth map from the Stereo Matching module as input. The pose estimate at the time of creation of the input data is required as an initial guess of the offset between the point clouds. The in-flight generated point cloud is matched to a reference point cloud, which was created previously on-ground from a DEM of the fly over area or of the landing site. First, the ICP searches corresponding points from the point clouds and second, estimates an optimal transformation that minimizes the distance between the correspondences. This is repeated until the optimization converges and a best guess of the pose estimate is achieved.
Hence, the Epipolar Geometry module, the Stereo Matching module and the 3D Matching represent a sequence of consecutive modules that provide an improved absolute position estimate. Due to the accuracy of the STR, only the position estimate is provided as the output to the Navigation Filter.

Shadow Matching: The Shadow Matching module provides an absolute localization in the lunar reference frame with the aid of the Binary Shadow Matching (BSM) algorithm described in [45]. The algorithm is based on the idea of using shadows on the lunar surface as landmarks. Given a camera image and the current pose estimate, the BSM extracts shadows from the image and creates descriptors for each extracted shadow. The descriptors are represented as one-dimensional binary vectors for memory and matching efficiency. These shadow descriptors are matched with reference descriptors which were previously computed on ground. In a final step, the matching result is used to compute an estimate of the absolute pose with a covariance. As an accurate orientation of the lander is provided by the STR, only the absolute position along with its covariance values are provided to the Navigation Filter.

Landing Site Detection: The Landing Site Detection module uses LIDAR input data to analyze surface characteristics for a possible landing. It models the ground area by raw point cloud data from the sensor and first returns a triangulated mesh surface with reduced information and thus faster processability. Second, the modeled terrain is searched for areas of low roughness and slope suitable for safe landing. Together with other high-level constraints that may come from the mission itself (e.g. vicinity to areas of interest), the identified landing spots are rated and inadequate targets are then suppressed. The ATON project includes an evaluation of the different heights at which any potential landing sites are detectable, which could have an influence on the overall planning of the landing mission.

Visual SLAM: Visual Simultaneous Localization and Mapping (SLAM) is used as an additional position estimator especially for cases where craters are not visible. In the ATON software architecture, it is regarded as part of the Navigation Filter module described below. It acts as an odometer and generally analyzes the motion characteristics of camera image features detected by the Feature Tracker module. Visual SLAM provides data on the relative motion of the camera and thus the SC. Such an approach is quite common in other robotics and virtual or augmented reality applications. Details of the algorithm are published in [4] with a focus on alternative aircraft navigation. However, this principle has been adapted here for the spacecraft scenario. Due to the nature of relative navigation, sole visual SLAM-based positioning is subject to error accumulation, but it is suitable for reducing the position drift of internal-based navigation principles when no other absolute position landmarks can be used.

Navigation Filter: The Navigation Filter module uses the output of the Feature Tracker, the Crater Navigation, the Shadow Matching and the 3D Matching along with the raw IMU, LA and STR measurements to estimate the true navigation solution. The Navigation Filter is based on high rate strapdown computation and a low rate error state Unscented Kalman Filter (UKF). The strapdown algorithm uses the IMU measurements to propagate the total navigation solution forward in time for each measurement. The low rate UKF estimates the error of the strapdown algorithm and corrects the propagated navigation solution based on the absolute position measurements from the other modules, the absolute attitude from the star tracker, and the altitude above the lunar surface measured by the altimeter. Additionally, the tracked image features are used in a visual SLAM algorithm to provide further position updates to the Navigation Filter.

3 Techniques and Technologies

This chapter summarizes the different techniques employed in the project ATON. This includes the different processing modules of the on-board navigation system. Further, simulation and software technologies have also been utilized to enable detailed simulations as well as the implementation and integration of a complex software. The following section provides a more detailed description for each of the elements.

3.1 Crater Navigation

Within the context of ATON, detection and matching of lunar impact craters in camera images is complementary to the 3D-Matching processing chain (cf. Sec. 3.3) and the Shadow Matching (cf. Sec. 3.4) approaches and yields the same class of measurement, which is a position value in the Moon-Centered Moon-Fixed (MCMF) coordinate frame. Conceptually, crater detection is attractive for its algorithmic simplicity, which follows from the physical rotational symmetry of most impact craters and their highly characteristic appearance under a wide range of external illumination conditions, as well as from their almost certain availability all over the lunar surface and their persistence over geological time frames.

A reliable crater detector therefore offers the potential to obtain measurements of landmarks in the MCMF coordinate frame during all mission phases, as long as the illumination conditions are within a certain range, the navigation camera is pointed at least partially at the surface, and there is a map of the craters available. In this section we will provide an overview of the algorithm design, which was devised in project ATON, in order to conduct these types of measurements. Figure 3.1 shows a flow diagram of the entire crater navigation algorithm chain.

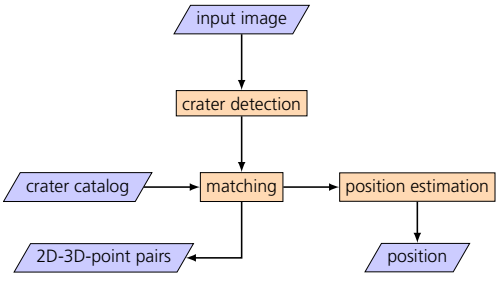


Figure 3.1: Crater navigation measurement cycle

3.1.1 Crater detection

Detection of craters in images of the navigation cameras is accomplished by extracting those Regions of Interest (ROI) of the image that exhibit the characteristic properties of an impact crater under direct illumination. These properties are:

- The ROI contains an area of significantly higher image intensity than the average intensity of its immediate exterior neighborhood
- The ROI contains an area of significantly lower image intensity than the average intensity of its immediate exterior neighborhood
- Depending on the crater size and the prominence of its rim wall, there may be two additional areas of above and below average intensity in close proximity outside the ROI
- These bright and dark areas are arranged in such a way that the crater interior dark area precedes the bright area with respect to the local direction of illumination

The characteristic appearance of a crater under illumination as described here is shown in Fig. 3.2, where Fig. 3.2a explains how terrain slope and illumination incidence angle progression produce a highly stable and predictable reflection intensity profile and Fig. 3.2b illustrates the resulting appearance with an example image.

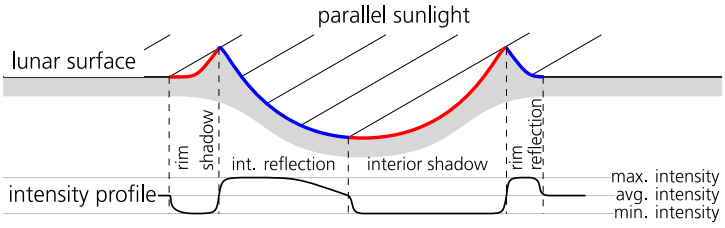
Some crater detection methods found in the literature are based on the detection of the crater rims in the image, instead of the detection of dark and bright image components as we propose. This approach was considered and rejected for a number of reasons:

- Crater rim detection requires image edge detection, which is by definition sensitive to scale. An algorithm based on edge detection therefore either makes an expensive scale space factorization indispensible or limits crater detections to a narrow scale (size) range.
- Crater rim edge pruning and grouping requires accurate a priori knowledge of the image illumination direction (denoted with s in Fig. 3.2b, right section).
- Crater rim edge grouping as proposed in the literature requires convexity tests that can fail in cases where crater rims do not appear convex (c.f. Fig. 3.2b, left section) and are sensitive to image noise and broken edges.

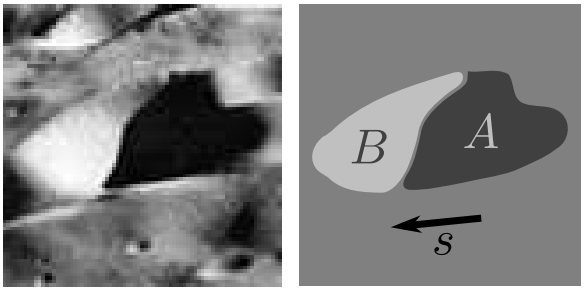
For a complete reasoning and an overview of the state of the art of edge detecting methods we refer to our publication [56]. The choice of adjacent bright and dark image components as detection targets is motivated in part by the shortcomings discussed above. The main advantages we identified are:

- The bright and dark image component detector is based on sorting and pixel proximity tests and does not rely on image intensity differentials. No scale space is required.
- The component grouping logic is based on relative size, arrangement and proximity. No convexity tests are required.
- The illumination direction that allows grouping of components is recovered from the data and is not required as an external input.

The method of detecting and grouping bright and dark image components can be reviewed in detail in our publication [56] and is briefly illustrated in Fig. 3.3. In summary, it is based on the concept of tracking connected components of binary images corresponding to a sequentially rising (or falling) threshold. A partial sequence of threshold images is shown in Fig. 3.3a, for the example of the left-of-center large crater in Fig. 3.3c. A tracking stability measure is applied to the centroid sequence of all connected components of these images, as can be seen in Fig. 3.3b, where each centroid

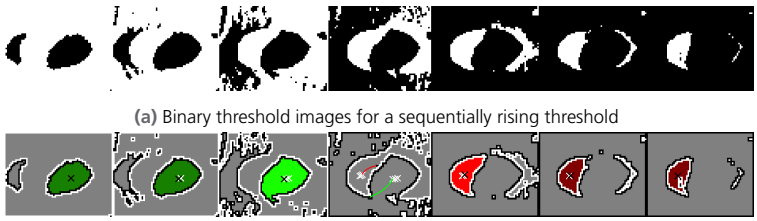


(a) Crater reflection intensity profile under illumination



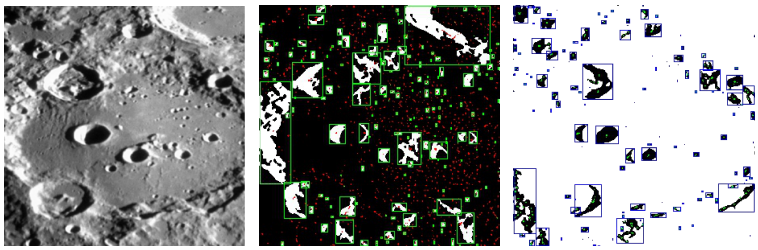
(b) Arrangement of crater interior reflection and dark areas with regard to the illumination direction; s denotes the direction of sun illumination

Figure 3.2: Characteristic appearance of impact craters under illumination



(a) Binary threshold images for a sequentially rising threshold

(b) Connected components within binary image with moving centroids



(c) Example image (crater Clavius) Credit: Conrad Jung, Chabot Space & Science Center

(d) Bright areas extracted with centroid traces in red

(e) Shadow areas extracted with centroid traces in green

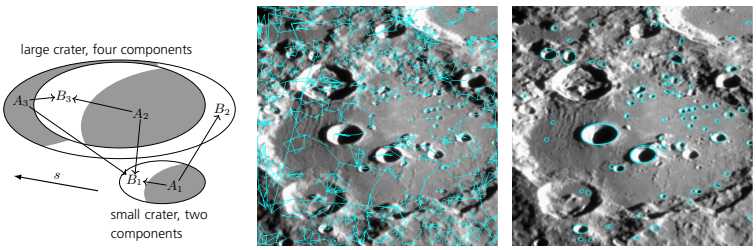
Figure 3.3: Extracting bright and dark areas from an image of the lunar surface

is marked with ×'s. Any connected component that is being tracked up or down through the threshold sequence is considered stable until its centroid moves disproportionately during threshold transition, as indicated by the arrows in the center field of Fig. 3.3b.

For the example image of Fig. 3.3c, all resulting stable dark components are shown in Fig. 3.3d and all bright components in Fig. 3.3e. Let a crater candidate be defined as the Principal Component Analysis (PCA) ellipse over the union of one bright and one dark component's constituent pixels. In order to recover the proper assignment between the sets of dark and bright stable components, an estimate of the image illumination direction is required.

The illumination direction can be inferred reliably by taking a statistic on the set of edges of a graph that connects every dark component with its two closest bright component neighbors of similar size and appropriate distance. Fig. 3.4a illustrates this idea, taking the example of one larger crater whose prominent rim ridge produces an additional bright and dark component, and one smaller crater where this is not the case. This oriented graph can be built for the whole example image of Fig. 3.3c, resulting in Fig. 3.4b. After pruning all edges from the graph that do not agree in direction with a mean-and-covariance-like statistic (c.f. [55]), and allowing exactly one assignment each between bright and dark components, PCA ellipses are fitted to the union of the assigned components' pixels. The set of remaining detected craters is then shown in Fig. 3.4c.

No further outlier rejection is performed on the set of detected craters. Rejection of mismatches between crater database and detection set, or database matches with false positive crater detections, are instead rejected by assessing their reprojection residuals during the pose estimation stage.



(a) Assigning closest two bright areas to each dark area within image section containing mixed large and small craters

(b) Graph of closest assigned bright and dark components

(c) Crater detections

Figure 3.4: Assigning bright and dark components and resulting crater ROIs

3.1.2 Crater matching

At any given instant of time, the navigation filter provides an estimate of the vehicle's pose. By extension, the pose of the camera within the MCMF coordinate system can be derived from its known physical alignment within the vehicle reference frame. By using the also known focal length of the camera, the crater catalog's moon-fixed entries may then be projected onto the camera image, resulting in an initial hypothesis about where known catalog craters should be visible in the image and about their size and orientation.

The goal is to assign as many as possible elements of the set of detected craters within the image to an element from the set of catalog craters that has been projected onto the image as described. The primary matching criterion is proximity (in image pixel coordinates u and v), consistency is assessed by comparing the relative magnitudes of the semiaxes of the respective ellipses, and the match is either rejected or not. Figure 3.5 illustrates the possible types of outcomes of matching detections (red) to projected catalog craters (blue).

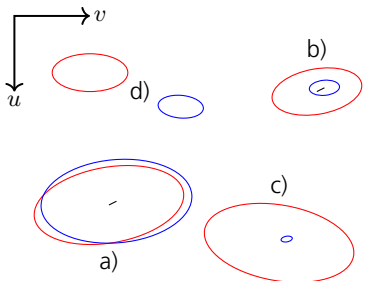


Figure 3.5: Detection and match classification

We consider four possible categories:

a) **True matches**. A detection is geometrically consistent with one of the projected crater catalog elements. Differences in their parameters are minor in relative terms and originate from detection errors or errors in the nagivation filter's state estimate.
b) **False matches**. A detection has been assigned due to its proximity to a catalog element with a significantly different size, but it is retained because a rejection threshold on the relative error of the semiaxes is not reached.
c) **Rejected matches**. A detection has been assigned by proximity to a catalog element with a size differing by an amount above a user-defined cutoff and has been rejected. This is the case when large craters intersect with small craters and the true match of either is not detected or known.
d) **Matchless outliers**. Dispersed detections where the crater detection algorithm either labeled something a crater which it is in fact not, or a correctly detected crater is not included in the crater catalog, as well as dispersed projected crater catalog elements which the crater detection algorithm failed to detect in the image.

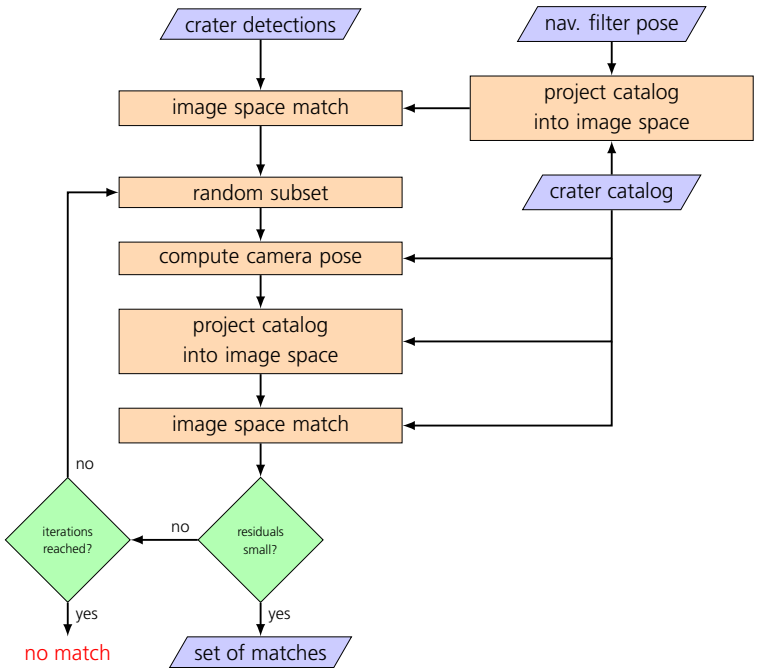


Figure 3.6: Match set outlier rejection

Image-space matching detections to catalog elements by proximity and rejection, based on relative parameter errors, leave a set of matches of the categories a) and b). The remaining false matches negatively influence the quality of any pose estimation based on the match set and therefore need to be reduced further or ideally be removed completely. To this end, we employ a Least-Median-of-Squares outlier rejection scheme over the reprojection residuals of hypothesized poses based on match subsets. Figure 3.6 shows in detail how from the initial set of image space matches between detections and projected crater catalog elements, random subsets are used to find a pose estimate not influenced by mismatches of type b) in Fig. 3.5. The testing terminates when the maximum number of iterations is reached or a predefined satisfactory residual bound has been reached. The match set with the least median of squared residuals is retained for the final pose estimation.

3.1.3 Position estimation

For the purpose of estimation of the vehicle pose, we employ a linear-time direct (non-iterative) combination of algorithms called Efficient Perspective-n-Point (EPnP) [49] and quaternion-based characteristic polynomial (QCP) [86] which directly yield an attitude quaternion for the transformation from moon-fixed to vehicle coordinates and a moon-fixed position of the vehicle.

These algorithms operate on sets of four or more correspondences between 2D image points (detected crater centers, image space) and 3D world points (catalog crater centers, moon-fixed). The pose estimation method is used primarily in reprojection residual-based rejection of matching outliers in small sets of correspondences (c.f. Fig. 3.6), where the full pose is required to reproject the crater catalog onto the image. The final position estimate that is supplied to the navigation filter as a measurement can either be the best (in a least median of squared residuals sense) posi-

tion estimate from this outlier rejection stage, or a solution considering all matches that can be made using this pose.

Theoretically, the attitude computed could also be fused in the navigation filter. However, the structure of the EPnP algorithm is such that the attitude is computed first as the relative rotation of two point clouds in camera coordinates and moon-fixed coordinates, respectively, and the actual moon-fixed camera or vehicle position is then derived from this rotation by considering the relative scales of the point clouds. This has the effect that errors in the position solution correlate with the errors in the attitude solution, a case that is difficult to properly treat in a Kalman-type navigation filter. We investigated this phenomenon in our publication [88]. For that reason, and for the fact that the star tracker's attitude measurement is more accurate by design, we only supply the crater navigation position estimate to the navigation filter.

3.1.4 Performance

As an optical method, the crater navigation performance depends strongly on the distance to the target. From this it follows that the error committed to the position measurement should be some fraction of the respective lines of sight to the detected craters. In order to confirm this assumption, simultaneous measurements of crater navigation position and line of sight distance to ground intercept can be used. Using data from the closed-loop flight experiment, Fig. 3.7 shows the error in the position measurement of the crater navigation module after normalization by the optical axis ground intercept distance, as measured by the laser altimeter. A histogram of these errors is given in Fig. 3.8. The variances of the error distributions show that the crater navigation can be expected to deliver a position measurement with an accuracy of about 1 percent of the line of sight, 1-σ.

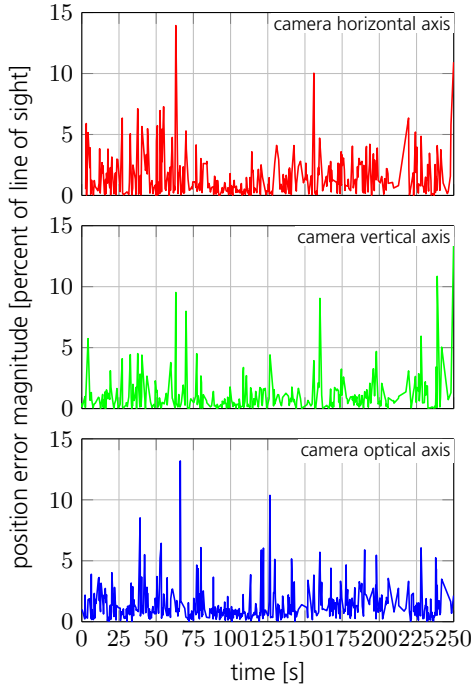


Figure 3.7: Crater navigation absolute position errors in closed-loop flight experiment

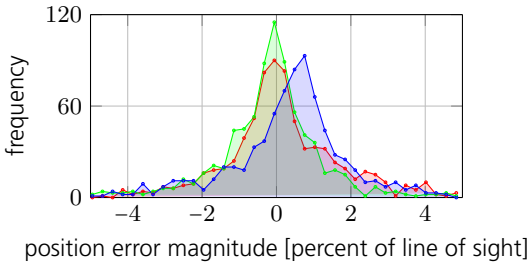


Figure 3.8: Crater navigation signed position error histogram in closed-loop flight experiment

This expected error variance is valid for cameras of similar focal length and similar image resolutions (i.e. about 35 deg horizontal/vertical field of view and 1 MPx). Other parameter configurations could be used to adapt effective performance to different mission requirements, trading off field of view (proportional to likelihood of a positive match) for accuracy of the position solution. Realtime capability is given with the method performing at a typical frequency of >1 Hz on an ARM core of a Zynq board, and an order of magnitude faster on an Intel i7 class CPU of the ATON navigation computer as it was used for the flight experiments (cf. Sec. 4.3).

3.2 Feature Tracking

Generally, feature tracking is understood as the extraction of 2D motion vectors from image sequences that describe the optical flow in suitable image regions. The feature positions and their change over time are then used as measurement input within the Navigation Filter (Sect. 3.7) for relative navigation. This requires visible motion, but it is independent of environmental knowledge, i.e. a map or catalog of known landmarks like craters. The scientific evaluation and implementation was already done

throughout the ATON project, however some recent extensions were implemented. The major elements of the feature tracking approach are published in [6].

3.2.1 Single Camera and 2D Tracking

A *feature* is a 2D pixel coordinate tracked over the sequence in a way that all occurrences of that feature are projections of the same physical 3D object point. Its change refers to visual motion, thus the characteristics of feature movements are an indicator for physical movements. Here, feature *tracking* is the image processing chain with the following tasks:

- Identification of multiple features in the image. The features must be visually traceable, i.e. significant contrast and texturing in the image region around the feature coordinate is required. Each feature is given a unique identifier which is kept over time.
- Identification of feature movements. There are two main approaches to doing this: finding image regions with maximum similarity to regions of a previous image (feature tracking), or independent classification of image regions and matching them between images of the sequence (feature matching). If visibility of an object is lost, the corresponding feature is lost, too, however it might be recovered when it becomes visible again.
- Feature monitoring and re-identification. This means checking whether features are still suitable for further tracking (or matching) since the visual appearance can change, and it involves enforcing a reidentification of new features if the amount becomes too low.

There is a large variety of feature tracking and matching algorithms. There is also extensive scientific work that assesses the different methods. In the ATON project, the core of the feature tracking module is a combination of feature identification based on high gradients in two axes [80] and the feature tracking based on image region similarity [53]. Implementations are based on the OpenCV library [66]. Compared to other (and newer) methods, the chosen method is fast to compute, which might be important on space-qualified hardware, and the method returns rather precise feature coordinates with sub-pixel accuracy. Since OpenCV only provides feature identification and tracking, some modifications have been implemented:

- Region search for feature identification. If the number of features is too low, the whole image does not have to be researched for new features. It is faster to find (rectangular) areas in the image where no features are present, and to search for new features within these areas.
- Tracing texture quality and feature distance. For each feature, the texture quality is measured by the variation of gray values around the pixel coordinate, and the distance to other features is measured. Features with low texture and with a distance too close to other features are discarded.
- Edge feature removal. As tracking quality is greatly lowered at the image border, features are discarded if they come too close (i.e. if its surrounding pixel region borders the edge).
- Local smoothing and outlier removal. In static environments (e.g. on the Moon), neighboring features are likely to have similar optical flow. If not, features with outlying movements are discarded as their identification or tracking is likely to be erroneous.

Given an image sequence as input, this implementation provides a set of features which is updated with every new image. It is automatically checked whether the number and quality of features are within specified thresholds. Image updates cause new positions of previously identified features, completely new features, or the removal of features when they are out of view or otherwise discarded. Each feature is described with:

- Unique identifier.
- 2D pixel position.
- 2D covariance matrix, here diagonal matrix with equal values.
- Identification and tracking confidence for internal purposes.
- Lifetime, increasing with every new tracking step.
- Timestamp which refers to the camera image timestamp.

Only valid features are written to the output. For external use, the feature coordinates are converted to normalized 2D camera coordinates using camera calibration parameters. The quality of feature tracking is assessed by the results of relative navigation.

3.2.2 Enhanced 3D Features using LIDAR Distance

The lander setup includes a ground-faced camera combined with a Laser range sensor (Flash LIDAR) that uses the same viewing direction. For ease of use and feasibility with currently available range sensors, a single distance measurement is combined with the camera images to get 3D features and further motion estimation without scale ambiguity. The sensor is considered as an altimeter which points about parallel to the optical axis of the camera (small displacements are considered and calibrated). It measures the slant range between sensor and ground surface.

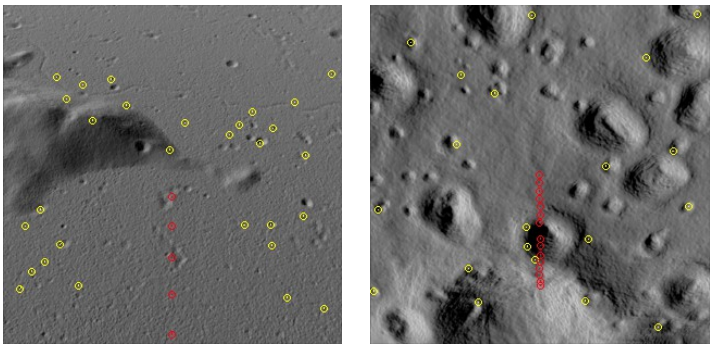


Figure 3.9: Tracked image features of a simulated Moon landing camera image sequence. Example images taken from 11 km height (left) and later from 300 m height (right). Features inserted with initial laser distance are marked red, the others yellow

Neglecting co-calibration uncertainties, an exact correspondence between image and distance measurement is achieved by the reprojection of the Laser range measurement onto the image plane. Except for very small distances, the laser beam is always reprojected to the same image pixel whose projection ray is parallel to the laser beam. It provides distance information for that specific pixel. Hence, 2D features at that coordinate can be provided with range and eventually with 3D coordinates of the object relative to the camera. In a simplified example with parallel sensor alignment, 3D information is available for the pixel at the optical axis. Fig. 3.9 shows some image examples of the simulated Moon landing sequence with 2D feature

tracking, and for a subset of images, an additional feature with 3D information is added and further tracked. As the vehicle moves, this 3D feature will also move and can be used as a 2D feature. Together with these new 3D features, existing 2D features can be extended with 3D information if they are very close to the (center) coordinate where distance is available. As a result, the feature tracker can provide 2D and 3D features to be used within optical navigation within Sect. 3.6. Further details on camera-LIDAR combinations with non-parallel viewing directions are presented in [5].

3.3 3D-Matching Processing Chain

The 3D-Matching Processing Chain's goal is to provide an absolute position estimate by creating and processing 3D data from camera images. It consists of three modules, the Epipolar Geometry module, the Stereo Matching module and the 3D-Matching module, which represent the three major processing steps. The 3D-Chain requires two-camera images, which are to be taken in short succession in order to provide enough overlap, and the initial camera pose estimate at the two recording positions. The Epipolar Geometry module is used to compute the exact relative transformation between the images which is required by the Stereo Matching module in order to provide a dense and accurate depth map. Finally, the 3D-Matching module matches this depth map to a 3D reference terrain model of the area currently overflown and provides an absolute position estimate to the navigation filter.

3.3.1 Epipolar Geometry Module

3.3.1.1 Motivation and Goals

The Epipolar Geometry module calculates a relative orientation (rotational and translational offset) of two images by improving given but inaccurate measurements from the navigation filter or other sensors. The translation cannot be determined completely. Only its direction in 3D space can be calculated. Its magnitude (i.e. the Euclidean distance between the centers of projection of the two images) cannot be determined accurately. Also, the absolute orientation cannot be determined accurately, but despite this, the resulting relative orientation is vital for the Stereo Matching and Local Depth Map Creation module to rectify the images and perform dense stereo matching. Due to the mentioned inaccuracies, the resulting depth map may contain systematic errors, which have to be resolved by external information (e.g. using the altimeter).

3.3.1.2 Methods and Functional Principle

The task is performed in two steps. First, corresponding image features are identified in the two successive images using the KLT feature matcher, which has been chosen for its low computational complexity. Sect.nd, a bundle adjustment is performed iteratively to determine the relative orientation and eliminate mismatched features at the same time.

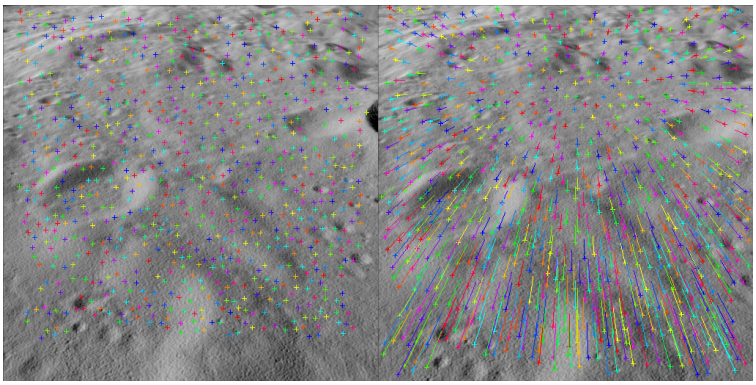


Figure 3.10: Feature tracking on simulated, subsequent images

By performing the bundle adjustment, the relative orientation parameters consisting of three translational and three rotational parameters (using Euler angles internally) are estimated from the observed motion of image features. As a significant part of the image features might have been matched wrongly, a RANSAC approach is necessary to filter out the outliers using a statistical approach. Finally, the estimated parameters can be used to perform dense matching via SGM. The drift of the scale of the translational parameters, which is not determinable in this setup, is not relevant for matching (but must be corrected afterwards, e.g. by the measurements of an altimeter).

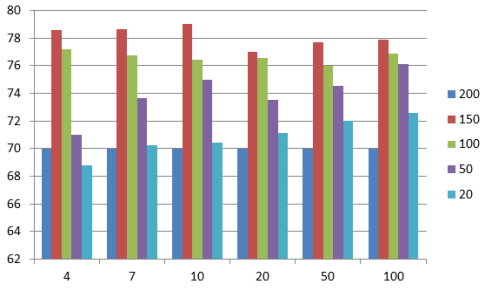


Figure 3.11: Mean success rate in percent, for minimum feature distances of 20 to 200 (colors) and 4 to 100 RANSAC iterations (column groups)

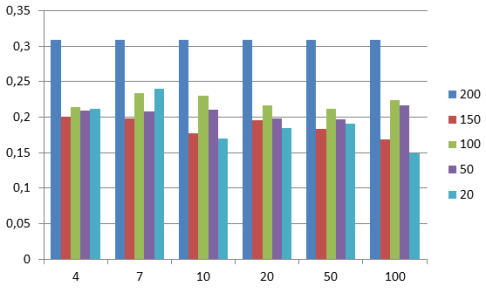


Figure 3.12: Mean epipolar error in pixels, for minimum feature distances of 20 to 200 (colors) and 4 to 100 RANSAC iterations (column groups)

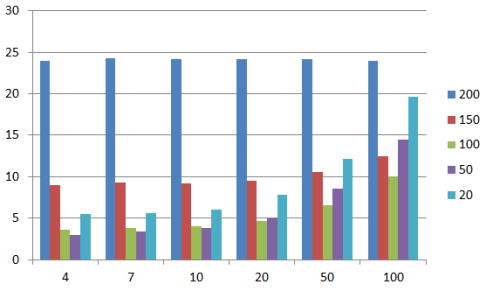


Figure 3.13: Mean computation time in seconds, for minimum feature distances of 20 to 200 (colors) and 4 to 100 RANSAC iterations (column groups)

3.3.1.3 Performance and Constraints

An extended performance analysis has been performed on simulated images. For the test, an artificial error of 1000 meters and 0.1 deg was added to the translation and rotation input parameters to simulate the errors of relative navigation via IMU between two images. The then calculated parameters were compared to the true, simulated values. The test was performed with different numbers of RANSAC iterations and different minimum feature distance values. Both of these parameters are known to influence the computational performance and accuracy of the results. In order to gain statistically relevant results for each combination of parameters, 30 runs have been performed: each with different random values for the artificial translation and rotation error, as well as RANSAC random values. Their results have been averaged. Figs. 3.11 through 3.13 show the results. Each group of columns represents a different number of RANSAC iterations. Within a group, each column represents a different minimal feature distance, in accordance with the legend on the right side of the referenced figures.

The results show that the performance varies significantly over the different parameters, but they do also show that there is a combination of parameters leading to good results with an acceptable computation time. With a minimal feature distance of 100 and 10 RANSAC iterations, the mean calculation time was about 8 seconds with the potential of reducing the computation time to 4 seconds with a still acceptable accuracy below 0.5

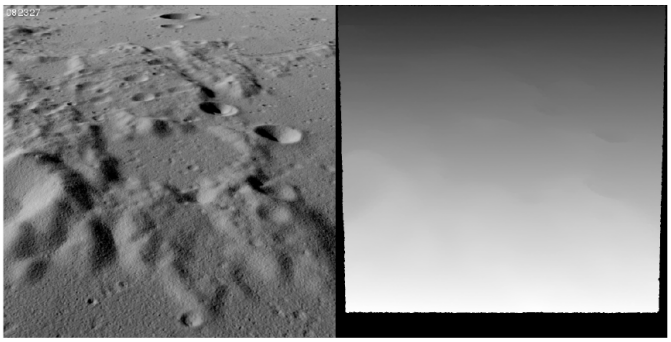


Figure 3.14: Example of a depth map (right) that was generated from a rendered view of the lunar surface. Distances are depicted by the brightness, i.e. closer areas are brighter than areas further away

pixels of epipolar error, known to be sufficient for SGM matching. The tests were performed on a standard PC with an i7 CPU from 2018.

3.3.2 Stereo Matching

3.3.2.1 Motivation and Goals

The purpose of the Stereo Matching module is to provide dense in-flight depth maps with a single camera by using the structure from motion technique [82]. By using two or more images from different viewpoints and by matching corresponding image points, it allows for the creation of a 2.5-D depth map. In the context of ATON, the depth maps produced in this manner shall serve as input to the 3-D Matching module (cf. Sect. 3.3.3).

We have used the SGM algorithm [29] which relies on images from passive cameras. Besides the lower power consumption, passive cameras also have a higher lateral resolution than other range sensing techniques such as a Flash LIDAR. Additionally, the SGM is a mature method that is known from robotics and aerial image processing for its high quality and dense depth maps. As Flash LIDARs are currently not available for planetary lander missions and as the accuracy of the subsequent 3-D Matching method depends on the resolution of the depth map, the Stereo Matching module is currently regarded as being a valid alternative in ATON.

3.3.2.2 Description of Method

The SGM uses two images, a known transformation between the images, and the intrinsic camera calibration as input, and it provides a depth map as output as shown in Fig. 3.14. Details about the SGM algorithm are provided in multiple publications [29–31].

Due to the parallax between the images, it is possible to triangulate the depth of corresponding image points that are visible in both images. Hence, the core task of the Stereo Matching module is to detect the correct correspondences in order to establish a relationship between the image points and to compute the disparity. A rectified pair of images is required, i.e. with parallel optical axes and parallel epipolar lines which are aligned to image rows. In this way, corresponding points appear in the same image row in the stereo images as shown in Fig. 3.15. Then the relationship becomes

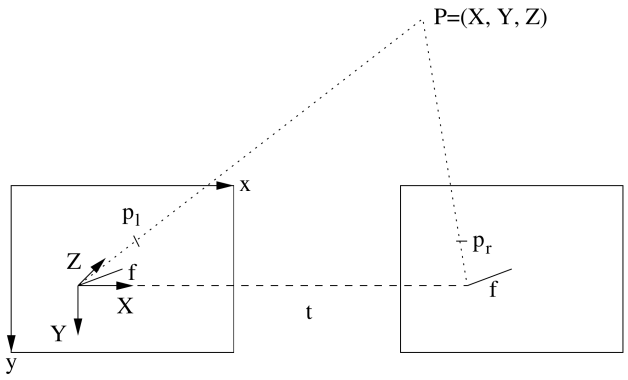


Figure 3.15: Geometry of a rectified stereo system. The object point P is projected to the corresponding image points p_l and p_r which are on the same epipolar line, separated by a baseline t

$$x_r = x_l + d(x_l, y_l), \tag{3.1}$$

$$y_r = y_l \tag{3.2}$$

where x_l, y_l and x_r, y_r are the corresponding image points in the left and the right image, respectively, and d is the disparity, i.e. the difference of image location for a point.

The actual depth Z is related to the disparity d as

$$d = f\frac{t}{Z} \tag{3.3}$$

where Z is the depth, f is the focal length and t is the baseline between two vantage points. Equations (3.1), (3.2) and (3.3) show that the use of an intrinsic camera calibration as well as the exact knowledge of the baseline t is required for the depth computation. This is the standard setup of a stereo vision system as it is used to derive the relations between the two images [82].

In order to achieve a high depth accuracy, the reprojection error Δp between two images must be less than 0.5 px.

The reconstruction error of a stereo system with horizontally separated images can be calculated based on Δp as

$$\Delta X = \Delta p\frac{Z}{ft}\sqrt{(t-X)^2 + X^2}, \tag{3.4}$$

$$\Delta Y = \Delta p\frac{Z}{ft}\sqrt{2Y^2 + \frac{t^2}{2}}, \tag{3.5}$$

$$\Delta Z = \Delta p\frac{Z^2}{ft}\sqrt{2}, \tag{3.6}$$

where f is given in pixels, t is given in meters and X, Y, Z are the coordinates of the reconstructed object point in the coordinate system of the left camera as shown in Fig. 3.15.

Instead of using two-cameras in a fixed arrangement, it is also possible to use just a single camera and to move it horizontally, i.e. perpendicular to the optical axis (in the XY-plane in the camera coordinate system as shown in Fig. 3.15). Given the knowledge of the exact motion from the first vantage point to the second vantage point of the camera, this will result in a configuration that is similar to the aforementioned two-camera stereo system. In the case of ATON, the single camera is moved forward as the lander moves on its trajectory and images from two consecutive points

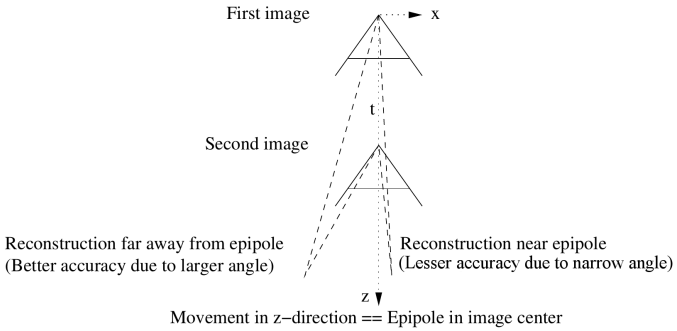


Figure 3.16: Mono camera system with a translation in z-direction, i.e. in the direction of view

in time, that provide sufficient overlap between them, are used. The same equations and concepts from the previously explained standard left-right stereo system hold for this first-second image stereo system. And as before, the region of overlap between the consecutive images defines the region in the resulting depth map that is filled with depth values. As before, it is necessary to use a pair of rectified images in order to lower the computational demands for the search for correspondences. This assumes a required maximum reprojection error Δp of less than 0.5 px which requires the knowledge of the relative transformation between the images with high accuracy as it is provided by the Epipolar Geometry module (cf. Sect. 3.3.1).

For the majority of the trajectory, the lander's movement is sufficiently horizontal with respect to the surface, such that the previously explained requirement for a camera movement perpendicular to the optical axis is achieved. However, a monocular camera also permits a movement in the direction of view, i.e. along the optical or z-axis. This situation is shown in Fig. 3.16. In this case, the reconstruction accuracy depends on the position of the corresponding points in the images. No reconstruction can be computed at the point in the direction of movement, i.e. the epipole. The reconstruction accuracy will be very low near the epipole and will be better further away from it. This can be expressed as

$$Z = \frac{x_2 t}{x_2 - x_1} \tag{3.7}$$

$$\Delta Z = -\Delta p \frac{Z(Z-t)^2}{X f t} \tag{3.8}$$

with x_1 and x_2 as the positions of projections on the image planes of the image above and below, f as focal length in pixels and t as vertical baseline. The reconstructed distance is Z, the distance error is ΔZ and the corresponding reprojection error Δp. An example of the described effect is shown in Fig. 3.17, where the first image was taken at 400 m, the second at 200 m and f was 2000 px. As can be seen, the part of the reconstruction that is in the direction of movement has a very high reconstruction error.

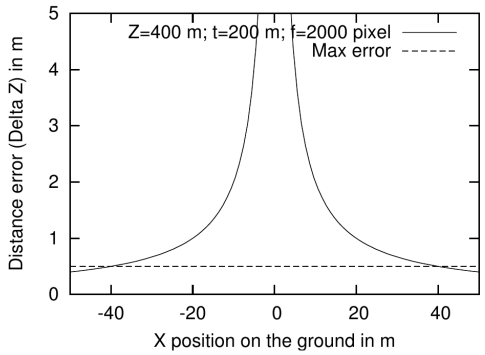


Figure 3.17: Example distribution of the distance error in the area that is covered by the lower image of a vertically moved mono camera system

In addition to the horizontal and the vertical cases (i.e. movement in XY- or Z-direction), there are of course mixtures of both. However, the general case is mathematically more difficult to describe and therefore omitted here.

An image-based approach for the determination of the camera movement cannot determine the length of translation, which means that the scale

of reconstruction is unknown. The scale may be determined by another measurement device, e.g. the LA may measure the distance to the surface. Alternatively, the reconstruction may be used without scale.

3.3.2.3 Performance and Constraints

For stereo matching, an image overlap of 75 % is a good starting point. The baseline for horizontal translation for an image with the width W can be computed by

$$t = \frac{WZ}{f}(1 - 0.75). \tag{3.9}$$

Due to the adaptive baseline, the reconstruction error will depend linearly on the altitude by the equation

$$\Delta Z = \Delta p \frac{Z}{W(1 - 0.75)}\sqrt{2}. \tag{3.10}$$

For a vertical movement which may occur at the end of the approach and landing phase, the situation is similar to the one shown in Fig. 3.17. Thus, a purely vertical movement should be avoided. Mixed movements would benefit from the horizontal translation component.

For an experimental confirmation, we used an image sequence of one of the simulated trajectories. The images have a resolution of 1024 px × 1024 px and a focal length of f =1406 px, which corresponds to an FOV of 40 deg. In the experiments, the baseline was chosen automatically, by starting with a manually defined image pair and adapting the time difference between the next image pair to a mean disparity of 100 px of the previous pair. In this way, the stereo base is automatically chosen according to the previous image pair. Fig. 3.14 shows an example.

The density of stereo matching was about 90 %, almost throughout the whole sequence. Exceptions are only those images with very dark shadows where depth values have been filtered out automatically. A comparison to the ground truth of the simulation data confirmed that the average depth error was between 0.1 % to 0.5 % of the altitude. Table 3.1 shows the Ground Sampling Distance (GSD), height error and frequency for certain altitudes and corresponding horizontal speeds. Although the required stereo baseline shrinks while descending, the stereo frequency is almost always about 0.1 Hz because the horizontal speed of the lander reduces in the same way. Thus, stereo matching using a moving mono camera

Table 3.1: Accuracy for stereomatching using a mono camera from 10 km to 2 km altitude

Altitude [m]	Horizontal Speed [m/s]	GSD [m]	Height Error [m]	Time Between Images [s]
10000	283	7.1	10 - 50	8.8
9000	251	6.4	9 - 45	9.0
8000	220	5.7	8 - 40	9.1
7000	189	5.0	7 - 35	9.2
6000	159	4.3	6 - 30	9.4
5000	129	3.6	5 - 25	9.7
4000	98	2.8	4 - 20	10.2
3000	67	2.1	3 - 15	11.2
2000	35	1.4	2 - 10	14.2

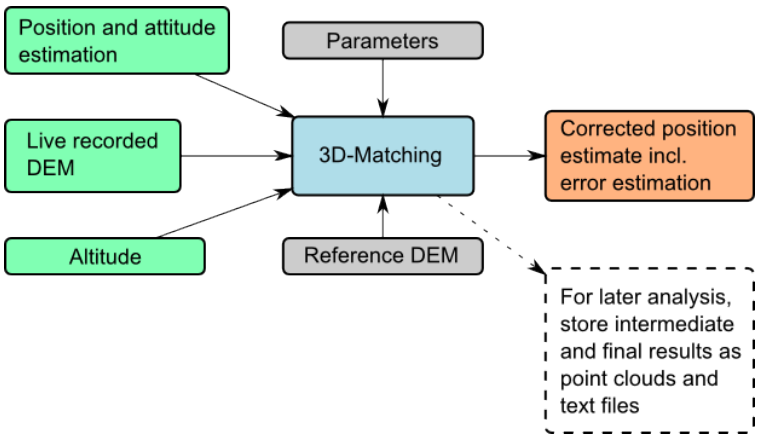


Figure 3.18: Overview of the inputs and ouputs of the 3D-Matching module in ATON

could be used for self- localization with the 3D-Matching module down to an altitude of approximately 2000 m. Below this altitude, the vertical component of the lander's movement becomes too large, which results in the case shown in Fig. 3.17.

3.3.3 3D-Matching

3.3.3.1 Motivation and Goals

The 3D-Matching module provides an absolute position estimate to the navigation filter. It matches a 3D reference model, for example of the landing area, to a 3D model from the Stereo Matching module or to a range image from a Flash LIDAR. As it makes use of 3D information, it is robust against illumination differences and can therefore reuse information from previous missions.

3.3.3.2 Description of Method

The 3D-Matching module is based on the ICP algorithm [9, 13, 75], which is a well-established algorithm in robotics and 3D data processing for the registration of 3D point clouds and which is known for its accuracy.

As shown in Fig. 3.19, the ICP matches two 3D point clouds, the object and the reference point cloud, for which it requires an estimate of their initial misalignment. For each point in the object point cloud, it tries to find a corresponding reference point. The displacement error for each correspondence is computed as the Euclidean distance between the points. The Root Mean Square (RMS) of all errors describes the overall displacement. In order to align the two point clouds, an optimization step tries to find the affine transformation T_a which leads to the smallest displacement error for the complete point cloud. The correspondence search and the optimization step are iteratively repeated until the RMS or the difference between the RMSs of consecutive iterations are below a threshold, or if the maximum number of iterations is reached.

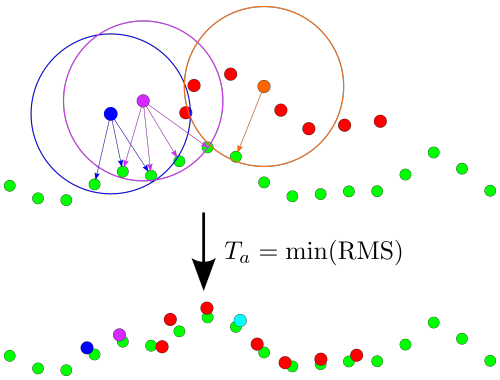


Figure 3.19: Concept of the ICP algorithm: Two misaligned 3D point clouds are registered to each other by finding the affine transformation T_a which minimizes the distances over all points. In an iterative manner, a corresponding reference point (green) is searched for each point of the object point cloud (red). For the set of correspondences, the transformation which minimizes the overall error or distances of all correspondences is computed. These steps are iterated until an optimal solution is found

In ATON, the reference point cloud was created out of DEMs of the relevant area, e.g. the area with the landing site. The object point cloud is either

given as a range image from a Flash LIDAR or as a depth map from the Stereo Matching module. The navigation filter provides the initial pose of the object point cloud which needs to be corrected.

In order to increase the speed of the correspondence search, which is the bottleneck of the ICP, the object and reference point clouds can be downsampled for the first few iterations. In these first iterations, the correspondence search is performed in a larger search radius, which would result in a large number of correspondence checks if the full resolution model were used. Once a coarse improvement of the initial pose is achieved, the resolution of the point clouds is increased and the search radius is decreased in order to achieve a high accuracy during the remaining iterations.

Additionally and due to the same reasons, the resolution of the reference point cloud is also adaptable to the possible resolution of the object point cloud. The latter depends on the distance between the camera and the surface, thus it is less for higher altitudes.

Once the ICP finishes, the difference between the newly found pose of the object point cloud and the initial pose is indicated by the optimal affine transformation T_a and it represents the offset of the lander from the planned trajectory. Adding the translational part of T_a to the initial pose provides the actual absolute position of the lander – which is the final result of the 3D-Matching module. Due to the usage of a high precision star tracker, the estimated attitude of the lander is not provided to the filter, although that would be possible.

3.3.3.3 Performance and Constraints

For the case of a lunar landing mission, the area covered by the FOV of the sensor is larger than the area of the reference point cloud during most of the descent. This changes only a short time before landing. With the applied implementation of the ICP algorithm, both cases can be handled.

Fig. 3.20 shows the accuracy that can be achieved with the 3D-Matching module over a range of different altitudes. The plot shows the result of a test run which used simulated Flash LIDAR data as input and a reference DEM of the landing site and its surroundings. The accuracy given in the plot denotes the error between the ground truth position of the lander and the one computed by the 3D-Matching module. The accuracy is given for all three coordinates separately and as a combination of all axes (pink line) in the MCMF reference frame. Here, the x-direction is approximately in the direction of the optical axis of the sensor.

As can be seen, the lateral accuracy of the 3D-Matching module (i.e. in the Y- and Z-direction in this case) provides a good accuracy but in the optical axis direction it shows larger errors. Because the laser altimeter provides accurate measurements in this direction, the larger error can be handled by the ATON system.

Figs. 3.21, 3.22 and 3.23 show examples of matching results for different altitudes at 15 km, 6 km and 1 km line of sight distance. As can be seen in these examples, matching is possible at high, medium and low altitudes. Green and red colors in the matching result, shown as point clouds, indicate whether a correspondence, i.e. a pair of points, is better or worse, respectively, than the final overall RMS. Despite a high number of correspondences with an offset larger than the overall RMS of the solution, the matching shown in Fig. 3.23 resulted in a high accuracy. Besides the

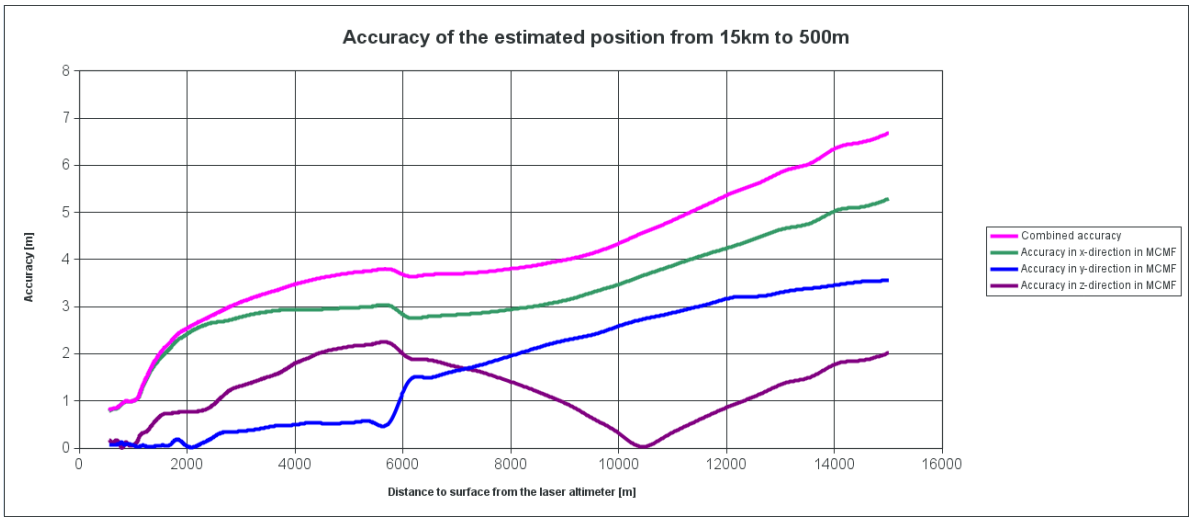
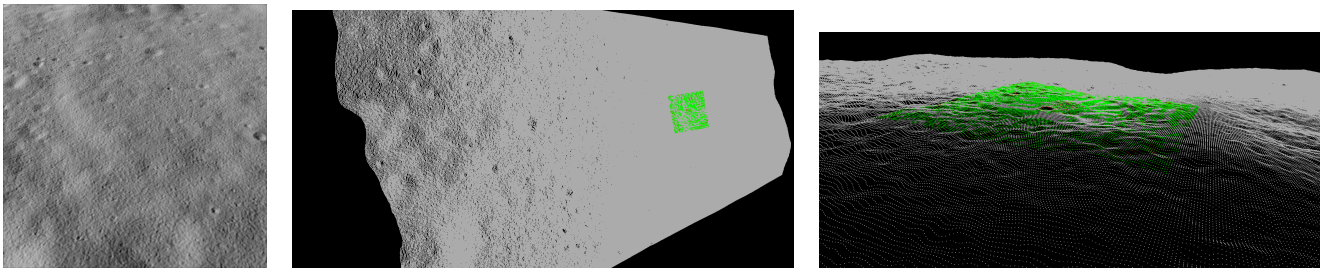


Figure 3.20: Accuracy of the 3D-Matching module with simulated Flash LIDAR range images for laser altimeter readings from 15 000 m down to 500 m

matching success, the results in Fig. 3.21 and Fig. 3.23 show that the ICP can deal with the case of the reference point cloud being either larger or smaller than the object point cloud, i.e. the FOV of the sensor.

In general, the experience from the project showed that given a good initial pose for the object point cloud it is possible to achieve highly accurate results for the position estimate at different altitudes. Due to the concept of the ICP, which is based on a local optimization, it is not robust against larger errors in the initial pose estimation. This can lead to a convergence to the wrong local minimum which cannot be detected by either the ICP algorithm or by the 3D-Matching module in its current implementation. In case a good initial pose estimate is provided to the 3D-Matching, it is well-suited to provide an accurate pose estimate refinement in order to achieve a very high accuracy, e.g. prior to the start of the landing site estimation.

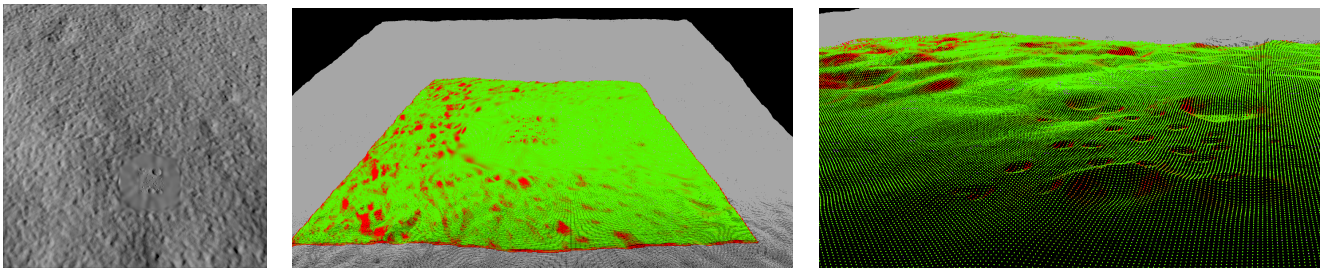


(a) Camera view at 15 km line of sight distance

(b) Overview of the matching result

(c) Detailed view of the matching result

Figure 3.21: Matching results at 15 km line of sight distance, where the landing site is barely visible. The Flash LIDAR point cloud (gray) and the reference point cloud (green) are shown after successful matching in an overview and in detail. The low resolution of the sensor data and the adapted resolution of the reference point cloud are visible. Green points represent successful point matches and red ones represent failed matches

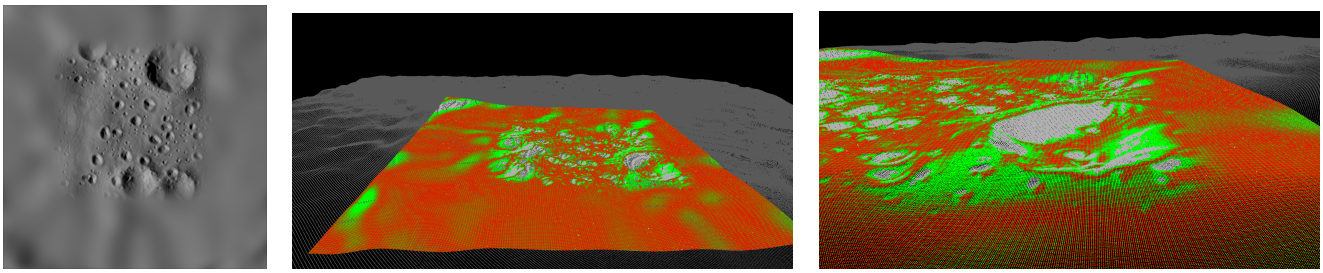


(a) Camera view at 6 km line of sight distance.

(b) Overview of the matching result

(c) Detailed view of the matching result

Figure 3.22: Matching results at 6 km line of sight distance with a well visible high resolution landing site. The Flash LIDAR point cloud (gray) and the reference point cloud (green) are shown after successful matching in an overview and in detail. The increased resolution of the point clouds is visible. Green points represent successful point matches and red ones represent failed matches



(a) Camera view at 1 km line of sight distance.

(b) Overview of the matching result

(c) Detailed view of the matching result

Figure 3.23: Matching results at 1 km line of sight distance were the landing site completely fills the FOV of the sensor. The Flash LIDAR point cloud (green-red) and the reference point cloud (gray) are shown after successful matching in an overview and in detail. Despite many point correspondences that exceed the quality criteria (red), the matching resulted in an accurate pose estimate

3.4 Binary Shadow Matching

The BSM algorithm is a vision-based absolute localization method for planetary landings first presented in [45]. It uses shadows on the surface of a celestial body as landmarks and can provide an absolute pose estimation in the body's reference frame, similar to the Crater Navigation presented in Sect. 3.1.

3.4.1 Motivation and Goals

The BSM is designed to be a robust and resource-efficient method that provides an accurate absolute position estimate in the final phase of the descent until shortly before the landing site selection phase and touch-down. As such the BSM is a method for vision-based self-localization which

- is able to provide an accurate absolute position estimate until shortly before touch-down,
- offers a sufficient level of robustness against deviations from the preplanned trajectory, both in time and space,
- is robust against illumination differences between reference image data and in-flight image data,
- is robust to some extend, against differences in resolution between the available reference data and the in-flight data,
- requires only images from a single passive camera in order to reduce the requirements on hardware and
- is by concept able to work efficiently, given the limited resources in terms of memory and computational power on-board a lander

In order to account for the last goal, the BSM tries to use as much of the knowledge available about the descent trajectory as possible in order to incorporate it into the reference data. By this, computationally intensive operations can be done prior to the more time-critical descent and landing phase. Originally, the BSM is designed for lunar landings, therefore some of its basic assumptions are related to the conditions specific to the Moon but the concept can be adjusted to other celestial bodies without an atmosphere, e.g. asteroids or comets.

3.4.2 Description of Method

In short, the BSM is a monocular approach with a feature descriptor for shadows that is robust against illumination changes between the reference and descent image data. The main idea is to make use of the abundance of shadows on planetary surfaces and to use them as landmarks as their appearance can be well predicted for reference purposes with the knowledge available from the mission planning.

The procedure of the BSM with the main processing steps is shown in Fig. 3.24. The overall BSM procedure is split in steps to be done on-ground prior to a mission and steps to be done on-board during the descent. On-ground, reference data for the matching step needs to be prepared, which includes the computationally intensive rendering of appropriate views (cf. Sect. 3.4.2.1). Besides this, the steps on-ground and on-board are similar. In both cases, gray scale images are converted into binary shadow images in order to extract the shadows from them. In a subsequent step the extracted shadows are described and stored as binary shadow descriptors.

During the descent, the reference and the on-line shadow descriptors are matched. The correspondences from the matching serve as input to the final pose estimation step, which provides an improved position estimate to the navigation filter. Each of these steps is explained in more detail in the following sections.

3.4.2.1 Reference Image Generation

Given a DEM of a planet's surface and the position and orientation of a camera at a specified time, it is possible to render the camera's view of the surface as it would approximately be seen during the mission. The knowledge of time, position and orientation of the camera also allows for the determine of the solar illumination of a specific area of the surface. By combining this information it is possible to predict the appearance of shadows on the planetary surface and use them as landmarks, especially on celestial bodies without an atmosphere such as the Moon or asteroids.

For the BSM, reference images are generated from DEMs of previous orbiter missions using the VTK[1] rendering tool. In our case, DEMs taken from the ATON lunar surface simulation or from the ATON high resolution landing site models [51] were used as input. These DEMs are geo-referenced, i.e. for each data point its real absolute position with respect to the planet's reference frame is known and can be used for the absolute localization.

Based on the planned trajectory for the descent, a virtual camera of the rendering tool is positioned in order to provide the same view as the lander's camera would have. The mission planning also provides the necessary information to set the correct direction and angle for the solar illumination of the scene.

[1] http://www.vtk.org/

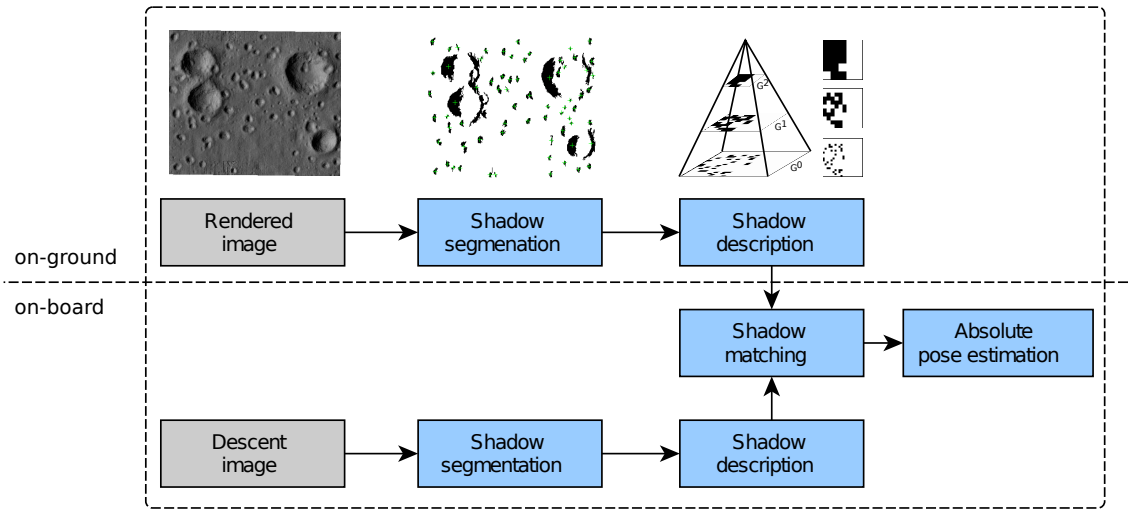
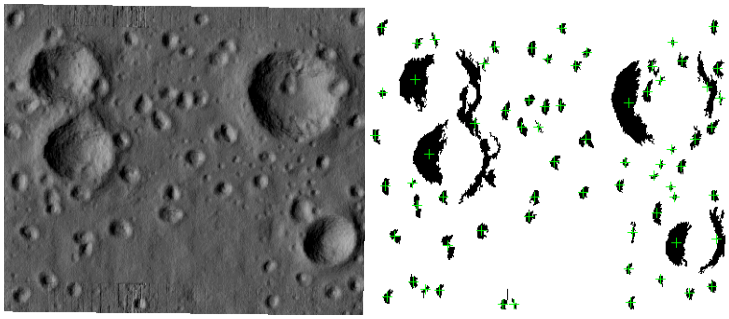


Figure 3.24: Overview of the BSM algorithm. The major steps on-ground and on-board are similar and only the input is different. A rendered gray scale image (gray) is the input to the on-ground shadow segmentation and shadow description pipeline; on-board it will be a gray scale image provided by the lander's camera. In the shadow matching step, the reference descriptors and the ones created on-board are matched in order to provide input to the final pose estimation step

In order to determine the shadowed regions in the rendering, we use the z-buffer algorithm that detects occluded areas in the scene. This is assumed to provide sufficiently correctly rendered shadows for our purposes due to the lack of a lunar atmosphere. This procedure is repeated with a time interval between consecutive images that guarantees a sufficient update rate of the pose estimation. The use of 3D information instead of image intensities for creating the reference views results in the desired robustness against illumination differences between reference and descent image data.

3.4.2.2 Shadow Extraction and Description

In order to extract the shadows from rendered images and from descent images, a binary shadow image is generated by applying a binary threshold to the gray scale images as in the example shown in Fig. 3.25a. This results in a black and white image as shown in Fig. 3.25b.



(a) Gray scale image **(b)** Binary shadow image

Figure 3.25: Input and output of the shadow extraction step, which converts a gray scale image into a binary shadow image. Here, the centroids of the shadows are marked in the binary shadow image (from [45])

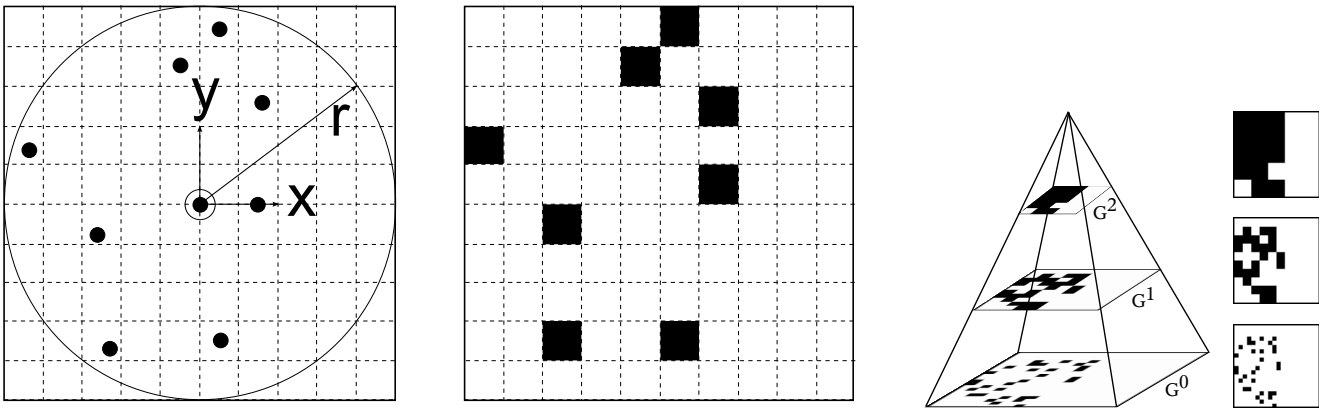
A binary threshold is automatically determined based on the image content with help of the so called gMET algorithm [35, 43]. It tries to maximize the information between the shadow pixels and the remaining pixels in the image. This results in the optimal threshold which segments the shadow pixels from the rest of the image. Experience has shown that applying a histogram equalization prior to the gMET algorithm improves the thresholding and the segmentation.

In the resulting binary shadow image, the contour of each shadow is determined and the centroid of the shadow, i.e. the center of mass, is computed from its first moments

$$c_x = \frac{m_{10}}{m_{00}}, c_y = \frac{m_{01}}{m_{00}}. \tag{3.11}$$

Shadows whose area is below a certain threshold are neglected for the following steps in order to achieve a certain robustness against noise and small differences between reference and descent images. Now each shadow is represented by the position of its centroid in the image. In the case of the reference images, this means that each shadow centroid is geo-referenced.

A robust description of a shadow landmark can not only consist of appearance information but also requires context information in order to make it more unambiguous. Hence, we chose to describe a shadow by the shadows in its neighborhood with a grid as shown in Fig. 3.26a, which can



(a) Projection of the shadow neighborhood onto a grid

(b) Discretization of the shadow neighborhood grid

(c) Pyramid of grid descriptors with varying resolution

Figure 3.26: Concept of the binary shadow descriptor and its generation. First the adjacent shadow centroids around one shadow are projected onto a grid, then discretized and finally stored in a pyramid of grids (Images from [45])

be matched relatively efficiently with respect to computational resources, thus the name Binary Shadow Matching. In order to describe a shadow as unambiguously as possible, a modified version of the grid algorithm of [67] is applied.

Here, all shadow centroids within a certain radius around a shadow are projected onto a grid as shown in Fig. 3.26a. The x-axis of the coordinate system points towards the closest shadow centroid and the y-axis is set such that an orthogonal coordinate system is achieved. In order to create a binary vector from the grid, it needs to be discretized as shown in Fig. 3.26b where only cells with a shadow in them receive a value of one, while the others are set to zero. The grid itself is stored as a one-dimensional column binary vector. Hence, each shadow's description relies on the relative position of adjacent shadow centroids in order to increase robustness and reliability of the descriptor.

In order to increase the speed and the robustness of the following matching step, a stack of m grids is created based on the original, discrete grid, similar to an image pyramid, as shown in Fig. 3.26c. On the top level, a downsampled version of the grid is stored followed by $m-1$ grid layers, each with an increased resolution, until the last level which contains the original, full resolution grid.

Using binary shadow descriptors results in a robust and at the same time memory-efficient representation for each shadow landmark. It also allows an efficient matching between the so called reference descriptors and the ones generated during the descent in order to account for the limited resources on-board. As discussed in [45], it is necessary to use a certain number of adjacent shadows in order to achieve a distinct descriptor. Hence, the neighborhood radius can either be fixed or adapted based on the number and distribution of shadows in the reference image in order to achieve a well-populated neighborhood.

3.4.2.3 Shadow Matching

If the grid pyramid contains (G_l) layers with $(l = 0, 1, \ldots, m-1)$, then the matching process starts at the top layer (G_m) which is the layer with the lowest resolution, i.e. l corresponds to the downsampling rate of the layer.

This grid is compared to the equivalent top level grid of all descriptors in the reference data. The (N) best correspondences are used for the matching on the next lower grid level, which is repeated until at the bottom layer (G_0) the final best match is determined. This procedure allows the search space to be quickly narrowed down at the beginning while achieving a high accuracy in the final steps.

The matching itself is a binary comparison, i.e.

$$\Delta \mathrm{d} = b_{ref} - b_{des} \tag{3.12}$$

if $(b_{[\cdot]})$ is the current binary vector of the reference grid or the descent grid, respectively and Δd is the Hamming distance between the two. But in order to account for the fact that often more shadows are present in the descent images than in the reference images, especially at lower altitudes, or for other cases of noise, a weighted binary comparison is applied; for details please refer to [45].

Outliers from the matching are removed with a RANSAC step, which iteratively determines the best fitting affine transformation (T_a) for the set of matched shadow centroids. Once the distance between a reference shadow centroid and a descent shadow centroid is below a threshold, the latter is considered an inlier or keypoint. In order to finish the RANSAC step successfully, a minimum number of keypoints must be present. Otherwise the RANSAC is stopped once the maximum number of iterations have been reached.

3.4.2.4 Position Estimation

The shadow matching establishes (n) correspondences between the 3D points (the reference shadow centroids) and the 2D points, i.e. the shadow centroids in the descent images. In combination with the intrinsic camera calibration this forms a Perspective-n-Point (PnP) problem that can be solved for the camera's position and orientation. Based on the fixed transformation between the camera reference frame and the vehicle's body reference frame, the PnP solution leads to the pose estimation for the lander.

The current implementation of the BSM includes two different approaches for the pose estimation. First, an iterative least squares approach with an iterative depth estimation [26] which was also used in [45], can be applied. It proved to be accurate, but required a relatively high number of keypoints for a stable and accurate solution (cf. Fig. 3.29). Hence, after an analysis of different pose estimation approaches, we decided to implement a second one, this time the closed-form approach for the P3P problem [46], which is the minimal form of the PnP problem with $n = 3$ point correspondences. It requires fewer keypoints, estimates the position and orientation in two steps and is known to determine the pose significantly faster than other state of the art methods [46].

3.4.2.5 Error Estimation

The confidence estimation is only performed for the estimated position, as the star tracker of the ATON system is assumed to provide an accurate orientation. The confidence estimation is based on the propagation of the reprojection error, which is common for the evaluation of the result of the

PnP problem. The 3D Points (X) are reprojected to the image plane with the estimated pose (t) with

$$y = f(x) \rightarrow d \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K(RX + t) \tag{3.13}$$

where (R) is the orientation of the lander, i.e. the rotation matrix, and (K) the calibration matrix of the camera. The residual (v) for each 2D-3D correspondence is the difference of the reprojected point (p_r) and measured 2D point (p).

$$v_i = p_r - p \tag{3.14}$$

The reprojection error is the sum of the residuals. We define the empirical reference standard deviation (s_0) to be

$$s_0 = \sqrt{\frac{vr^T Pvr}{r}} \tag{3.15}$$

where (r) is the redundancy, i.e. the number of 2D-3D correspondences. The reprojection error is propagated to the unknown position following the general law of error propagation. Solving equation Eq. (3.13) for the translation (t) holds:

$$x = f^{-1} = G(y) \rightarrow t = K^{-1} d \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} - Rx_w \tag{3.16}$$

Hence, the standard deviation for the translation in x, y and z is

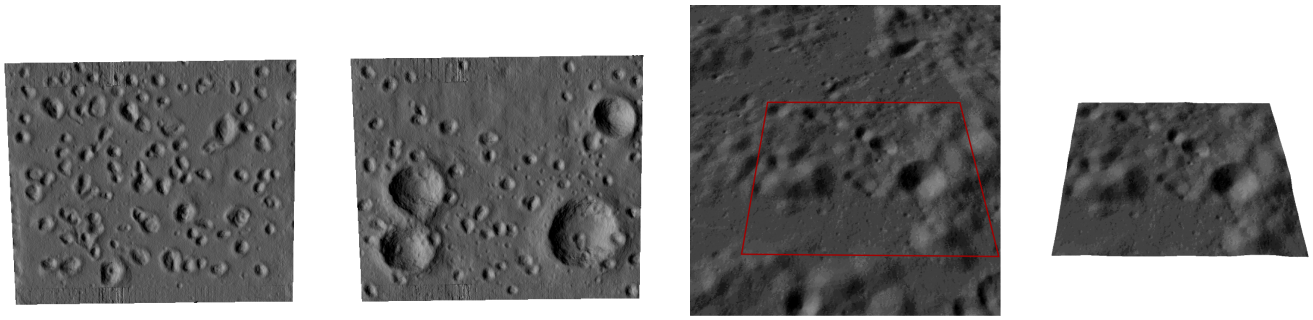
$$\begin{aligned} \Delta t_x &= s_0^2 \cdot \sum_{i=1}^{n} \sqrt{\frac{\partial G_x}{\partial u_i}^2 + \frac{\partial G_x}{\partial v_i}^2 + \frac{\partial G_x}{\partial d_i}^2} \\ \Delta t_y &= s_0^2 \cdot \sum_{i=1}^{n} \sqrt{\frac{\partial G_y}{\partial u_i}^2 + \frac{\partial G_y}{\partial v_i}^2 + \frac{\partial G_y}{\partial d_i}^2} \\ \Delta t_z &= s_0^2 \cdot \sum_{i=1}^{n} \sqrt{\frac{\partial G_z}{\partial u_i}^2 + \frac{\partial G_z}{\partial v_i}^2 + \frac{\partial G_z}{\partial d_i}^2} \end{aligned} \tag{3.17}$$

As the final result of the BSM algorithm, these values are used as confidence values and are returned to the navigation filter together with the position estimate. Again, due the use of a high precision star tracker, the estimated attitude of the lander is not provided to the filter.

3.4.3 Performance and Constraints

As some of the main assumptions of the BSM, for example the lack of an atmosphere and the known direction of the Sun during the descent, are hard to ensure in a terrestrial environment, most of the tests were performed with data from the simulated lunar scenario. A small test series was also performed with real data from one of the flight campaigns.

For the lunar scenario tests, three different landing site models, shown in Fig. 3.27, were used to create the reference data and the descent image data. The trajectory of scenario 6 (cf. Table 2.6) was used to simulate the lander state and the solar illumination. Tests with site 1 and 2 were performed from lower altitudes with a line of sight distance between 4 km



(a) Landing site 1 **(b)** Landing site 2 **(c)** Landing site 3 surrounding area and the cut-out that was used for reference data

Figure 3.27: Views on DEMs of the landing sites used for the performance evaluation of the BSM. Landing sites 1 and 2 are based on models from [51] and landing site 3 is taken from the lunar simulation

to 1 km in order to represent the end of the trajectory. The tests with site 3 were performed at mid-level altitudes, i.e. from 36 km to 12 km distance to surface in order to evaluate the performance at an earlier phase of the descent. The three sites show different topological characteristics in order to achieve a certain variety in the tests. See also [45] for more details about the experimental setup.

3.4.3.1 Robustness against Deviation from the Planned Trajectory

The BSM was tested for positional deviations up to 400 m from the ideal trajectory, i.e. 10 % to 1 % of the distance to surface, depending on the site. In order to test the robustness against changes of the sun incidence angle, the time of arrival of the lander was delayed up to 45 min which is assumed to be beyond any possible delay of the lander with respect to a planned trajectory. A delay of 45 min results in a change of the sun incidence angle of approximately 0.4°.

Shadow matching was successful at all three landing sites. For landing sites 1 and 2, more than 90 % of the shadows could be matched correctly, despite any deviations. At landing site 3, shadow matching was more challenging due to the lower number of shadows in the images, with matching rates of only 40 %. A low number of shadows can lead to sparsely occupied shadow descriptors, thus increasing their ambiguousness and the potential of mismatches.

As with the matching, the pose estimation of the BSM also proved to be robust against both kind of deviations. As a result, the BSM is able to provide position estimates with an average error of less than 1 % of the line of sight distance. The error is largest in the direction of the optical axis, which does not pose an issue as it is covered by the measurements of the laser altimeter. In the case of the plot in Fig. 3.28a the optical axis is in the x-direction and in the case of the plot in Fig. 3.28b it is in the z-direction.

Depending on the applied pose estimation method, the BSM is affected by the number of key points, i.e. the number of successfully matched shadows. In case of the iterative pose estimation method, a value of 70 key points turned out to provide on average accurate results with a small standard deviation over numerous test images regardless of the altitude. Fig. 3.29 shows this behavior for the extreme case of the maximum tested deviation of 400 m. A smaller number of key points is also possible, but

depending on the image content and the deviation it can increase the inaccuracy. Additionally, a more or less uniform distribution of the shadows over the image is advisable in order to reduce errors of the pose estimation.

The BSM, even in an early non-optimized implementation, is able to provide an absolute position update with a frequency of less than 0.1 Hz. Especially the pyramidal descriptor as shown in Fig. 3.26c increased the speed but also the accuracy significantly, whereas the RANSAC-based outlier removal requires the majority of the computation time.

3.4.3.2 Robustness against Image Resolution Differences

As the BSM's reference data is generated from DEMs of the surface, which can be of much lower resolution than the images from the lander's camera, its robustness against differences in resolution between reference data and in-flight data was investigated. This is of importance in order to qualify the BSM for use at lower altitudes.

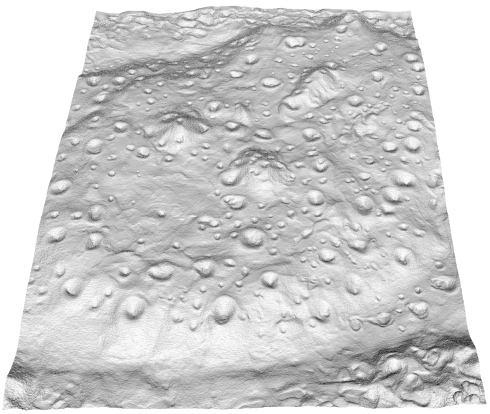
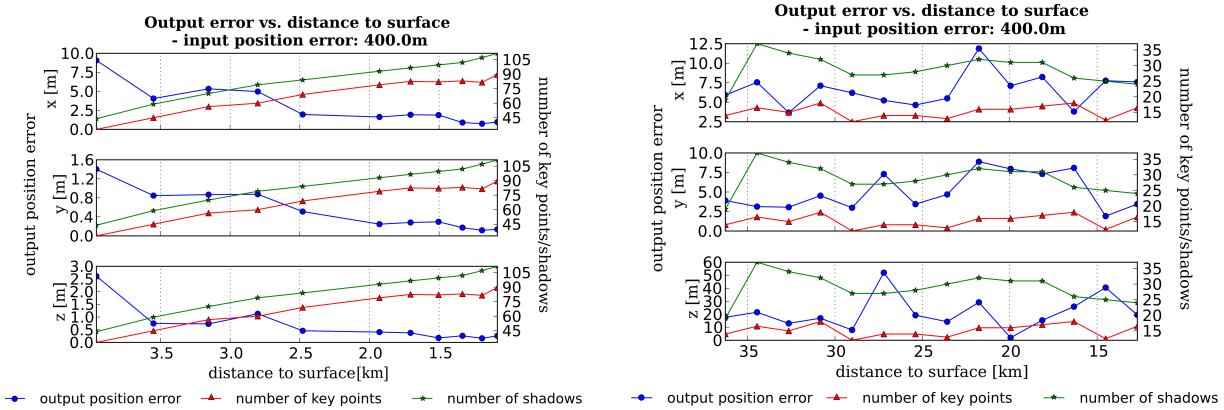


Figure 3.30: Model of a landing site on a ridge between two large craters. The model was used additionally to landing sites 1 and 2 for the resolution tests

Additionally to landing sites 1 and 2 (cf. Fig. 3.27), for this experiment a third landing site was used. Fig. 3.30 shows this model, which represents a ridge between two large craters.

The resolution of the three landing site DEMs was decreased up to 12 times, from 0.6 m/px to 7.2 m/px, as shown in Fig. 3.31. For each of the low resolution DEMs, BSM reference data was computed and then matched with full resolution data, i.e. with data with 0.6 m resolution as shown in Fig. 3.32.

The examples from the matching in Fig. 3.32 show that lowering the resolution leads to cases where smaller shadows disappear, which can be handled well by the BSM's weighted matching scheme. Additionally, some shadows might split up or merge when the resolution is lowered, which can lead to a number of mismatches. The case in Fig. 3.32c shows that at very low resolution levels, large shadows can emerge. Experience from



(a) 0.6 m/px

(b) 2.4 m/px

Figure 3.28: The error of the position estimated with the iterative pose estimation for landing sites 1 and 3. Error of position, number of shadows and number of key points with respect to the line of sight distance are plotted. The left axis of ordinate denotes the output position error (blue, dot), the right axis of ordinate denotes the number of shadows (green, star), and the number of key points used to estimate the position (red, triangle). Please note that the performance of the BSM for landing site 2 was similar to landing site 1

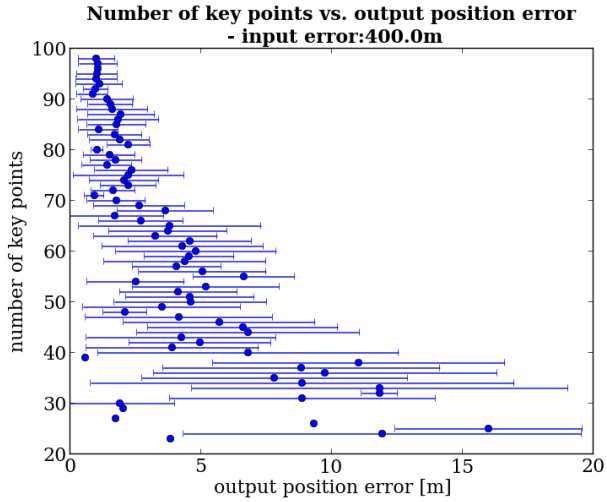
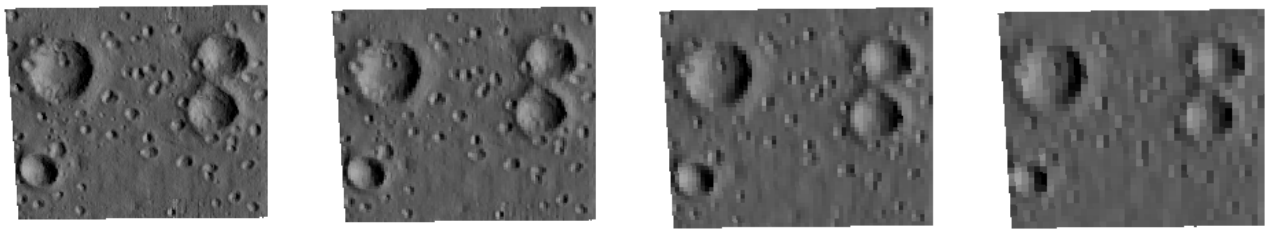


Figure 3.29: The number of key points versus the error of the estimated position for landing site 1 for a position drift of 400 m at any tested distance to the surface. The blue points mark the mean value of the position error and the whiskers show the standard deviation $\pm\sigma$. Points with no whisker mean that only one pose estimation was performed with the specified number of key points

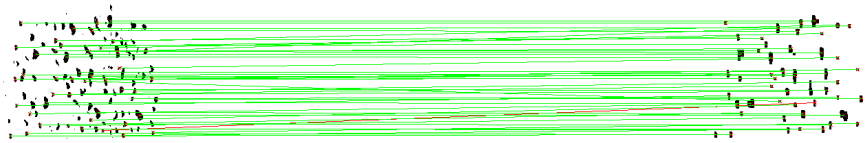
testing with the BSM shows, that in scenes that mainly consist of few large shadows, the matching is very likely to fail. It might be possible to identify such cases beforehand during the mission analysis phase in order to avoid operating the BSM under unfavorable conditions and with the risk of using significantly degraded position measurements.

Nevertheless, the plots in Fig. 3.33 show that the BSM is able to handle the resolution differences well and that it can provide very high matching rates (aka. true positives) for the majority of the cases. Even a maximum rate of mismatching between 10 % and 30 % can still lead to an accurate pose estimation given the experience from the deviation tests.

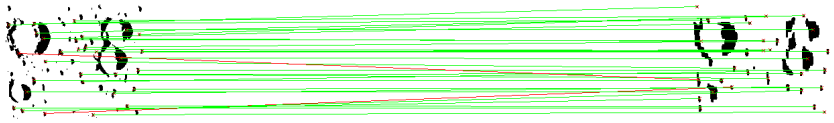


(a) 0.6 m/px **(b)** 2.4 m/px **(c)** 4.8 m/px **(d)** 7.2 m/px

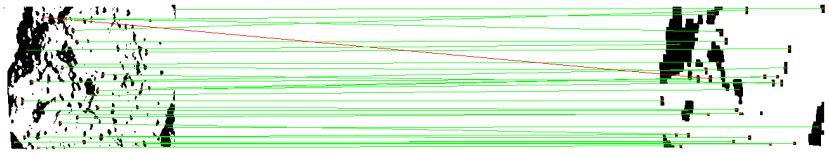
Figure 3.31: Landing site 2 with the different resolution levels used for the experiment, from the original resolution of 0.6 m/px to 7.2 m/px. The resolution decrease shown is 4, 8 and 12 times



(a) Landing site 1

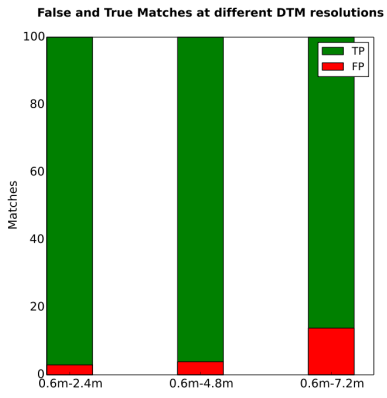


(b) Landing site 2

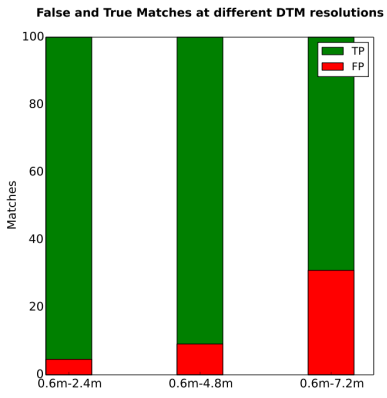


(c) Landing site on ridge

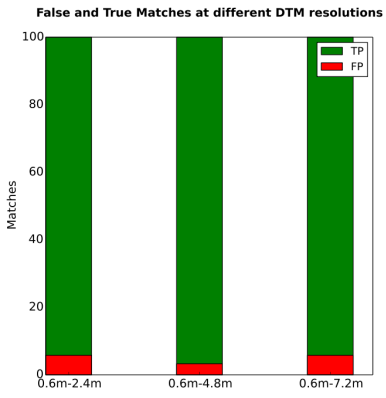
Figure 3.32: Examples of the shadow matching for the three different landing sites. The full resolution, i.e. in-flight image, is shown on the left, whereas the lower resolution reference images is shown on the right. The resolution of the reference images was 12 times less than the original resolution, i.e. 0.6 m/px versus 7.2 m/px



(a) Matching rate landing site 1



(b) Matching rate landing site 2



(c) Matching rate landing site on ridge

Figure 3.33: Performance of the shadow matching for the three different landing sites with resolution differences. TP (green) stands for true positives, i.e. correctly found matches and FP stands for false positives, i.e. mismatches. The values are given as a percentage of the overall available shadows in the reference images

3.5 Landing Site Detection

3.5.1 Surface Representation

The 3D surface modeling based on 3D LIDAR data is one source for three-dimensional modeling of the lunar surface [84] during the final landing phase. The creation of the model begins with the operational phase of the LIDAR, which starts about one kilometer above the lunar surface.

3.5.1.1 Modeling Method

The surface modeling method is an iterative irregular Delaunay triangulation, which produces a 2.5D surface relief of the Moon for a locally restricted area. During the triangulation, the modeling takes the shape of the surface into account in order to achieve a reduction of the original point cloud volume. Thus a reduction of the computational and memory requirements of the modeling process can be achieved. The implementation of the Delaunay triangulation is based on an incremental approach. In contrast to other approaches (e.g. flip or divide and conquer), this method allows the construction of the grid network with a point quantity which is not completely known from the beginning.

Basically the Delaunay triangulation is a 2D method. In the conversion as a 2.5D variation on the basis of Cartesian coordinates, the triangulation takes place in the x-y plane. The z-scalars of the 3D LIDAR data have no influence on the triangulation. They are used after the triangulation to construct the elevation relief. The 2.5D relief reflects the lunar surface, but it cannot represent structures like tunnels or overhangs. These types of structures do not occur naturally on the Moon, so the simplification does not lead to any disadvantage. Fig. 3.34 shows an exemplary lunar surface section of the triangulated LIDAR data.

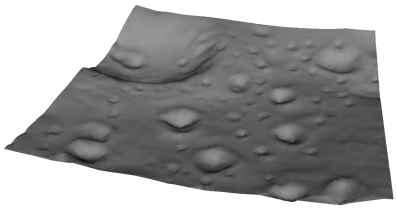
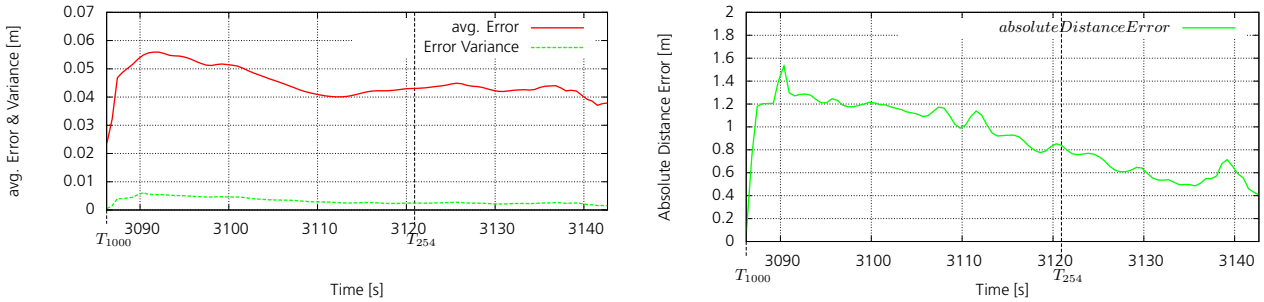


Figure 3.34: Representation of the lunar surface with a Delaunay Triangulation based on simulated LIDAR data

3.5.1.2 Performance

The geometric error between original point cloud $A : (a_1, a_2, ..., a_i), a_i \in \mathfrak{R}^3, 1 \leq i \leq m$, which is generated by the LIDAR, and the triangulation is used to verify the accuracy of the resulting triangulation. Triangulation is a simplification of an original point set and does not necessarily contain all points of it. Therefore, the original points are mapped to the surfaces of the triangulation and not to its vertices. This conversion allows a quality estimation of the simplification process.

The geometric error is determined by the orthogonal distance between the surface $F_k : (f_1, f_2, f_3), f_j \in \mathfrak{R}^3$ and a measurement a_i of the point cloud. The identification of the surface F_k, which containing a_i, is based on the location of $a_i^{'} \in \mathfrak{R}^2$ and the $a_i^{'}$ surrounding polygon $F_k^{'} : (f_1^{'}, f_2^{'}, f_3^{'}), f_j^{'} \in \mathfrak{R}^2$. $a_i^{'}$ and $F_k^{'}$ located at the x-y-plane of the triangulation. For the evaluation of the triangulation the ATON test data of scenario 6 (see Table 2.6) were used. The test begins at $t = 3086.2\overline{3}$ s. At this time T_{1000}, the lander is about one kilometer above the ground, which is the maximum range of the LIDAR. Another important time T_{254} is around $t = 3120.9\overline{3}$ s where the lander is about 254 m above the surface. At this level, the resolution of



(a) Avg. error and error variance between triangulated surface model and original point cloud per LIDAR scan.

(b) Max. abs deviation between triangulated surface model and original point cloud per LIDAR scan.

Figure 3.35: Deviation between triangulated surface model and an original point cloud

the LIDAR is approximately six measurements per meter, which is considered as necessary to achieve a sufficient assessment of the ground for a subsequent landing site evaluation.

For the evaluation of the triangulation, the average geometric error and the variance of the geometric deviations between the original point cloud of the LIDAR and the triangulation were determined using the described method.

The graphs in Fig. 3.35a show the results for the average geometric error as well as the variance of the geometric deviation. The average error of the surface model is, after a short phase at the beginning of the recording period, below the the required roughness $R_{max} = 0.05\,\text{m}$. Similarly, the low error variance suggests that the number of outliers is limited.

In addition to the average errors and their variance, the absolute geometric errors per LIDAR image were evaluated and presented in Fig. 3.35b. This shows that the maximum absolute deviation between the initial data and the model surface is 1.5 m. It can also be seen that it decreases with decreasing distance to the lunar surface. Finally, the maximum of absolute error is about 0.8 m at T_{254} and 0.5 m at touchdown.

The scope of data reduction by the triangulation depends on the distance of the LIDAR to the lunar ground. The triangulation uses between 1 % and 8 % of the input data to generate the surface model. Based on the previously described evaluation results (average and absolute error), it can be concluded that the number of measured data used is sufficient for the desired surface accuracy.

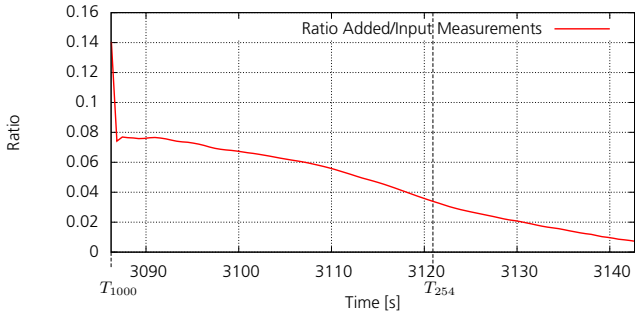


Figure 3.36: Ratio between point cloud size and the number of vertices used for the triangulation

3.5.2 Landing Site Validation

The aim of the validation is to identify and evaluate landing sites for a safe automatic landing of an unmanned vehicle on the Moon. Identification and assessment of the landing sites are based on the online-generated terrain surface model used in the simulation which depicts the lunar surface as an irregular grid by means of a 2.5D Delaunay triangulation. For the evaluation of the given grid surface as well as the differentiation of the regions suitable for landing, it is necessary to evaluate individual surfaces or combinations of several surfaces by means of criteria such as slope, roughness, and size of the landing zones [77]. In addition, scientific objectives, the remaining fuel or the surface strength can be considered [79]. The size and shape of the terrain on which a landing can be pre-planned and performed safely is based on the shape of the orbit. For a simplified assessment of the landing site size, the projected base area of the lander is conservatively represented by a circle.

3.5.2.1 Algorithm

Based on the irregular structure of the Delaunay network, it cannot be assumed that a single grid cell (area) of the surface model has a sufficient size to allow a safe landing. For this reason, areas which generally allow a landing based on their inclination and roughness are grouped into landing zones. Landing zones are limited in size only by the properties such as inclination and roughness of the containing areas and can contain multiple landing sites. Therefore, the use of a landing zone as an exclusive site leads to the neglect or downgrading of other potential landing sites. The aim is a separation of the landing zone into several permissible landing sites. Based on the simplified lander geometry projection and the polygonal shape of the landing zone, the algorithm from [21] identifies all circles with a radius r_{min} which lie within the landing zone and defines them as possible landing sites. In order to obtain initial solutions for the possible centers of the landing sites, the landing zone polygon is discretized with a point grid. An example of the described approach is presented in Fig. 3.37. The vertices of the polygon P are blue and the grid points are black. Furthermore, Fig. 3.37 shows three circles with the radius r, which was previously defined as the smallest possible landing site radius. For the two green circles, the minimum landing area criterion is fulfilled. Their center thus represents two permissible landing sites. Contrary to this, the red circle intersects with the boundaries of the polygon and thus does not meet the minimum size required.

With the presented process, several landing sites can be found. In order to decide which is the best alternative or to consider further requirements, it is necessary to sort the detected landing sites. A fundamental sorting is done depending on the roughness and inclination of respective landing sites. In addition to the basic sorting parameters, further parameters can be used, e.g. distance to a scientifically important target location or a maximum distance to the nearest obstacle.

The evaluation of these criteria is not based on the characteristics of the individual surfaces that form a landing site, but rather on a regression plane which is based on the corner points of the individual surfaces of the respective site. An example of a sorting shows Fig. 3.38 which visualizes a distance-based sorting towards a preferred landing location (gray circle).

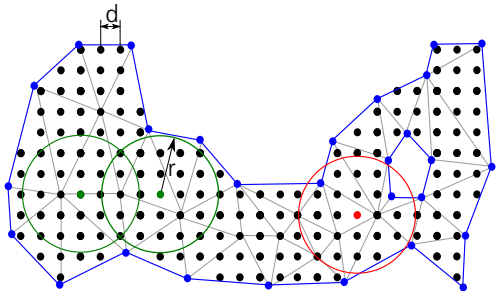


Figure 3.37: Discrete rasterization of a polygon to identify centers of possible landing sites

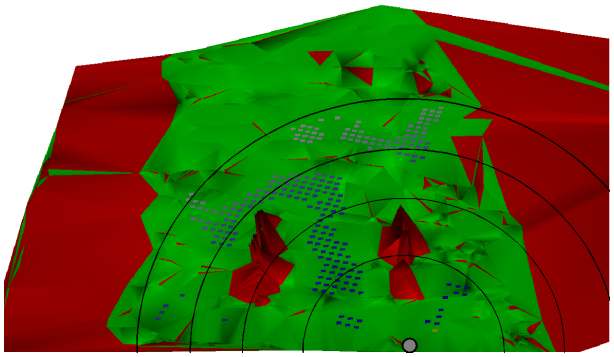


Figure 3.38: Sorting landing sites (coloring) based on the distance to a preferred region

It can be seen that only the landing zone (green area) contains possible landing sites. The choice concerning which grid faces are part of the landing site is based on their roughness and inclination. The dots inside the landing zone represent the identified landing sites and their color-coding provides the distance to the target region. Darker colors represent a lesser deviation to the target than brighter colors.

3.5.2.2 Performance

An assessment of the quality of a single landing site is not possible given the current state of the work since reliable knowledge about parameter limits are missing but necessary for a safe landing. References in the literature to determine which thresholds are appropriate provide as yet inadequate information for judging whether landing sites created using these parameters give a realistic picture of the actual landing sites.

Based on scenario 6 of the ATON data set (Table 2.6), a quantitative evaluation was done to identify the maximum number of landing sites for different heights of the lander above the lunar surface. Promising parameters for defining a landing site are a maximum surface slope $delta_{max}$ = 10 deg and a maximum surface roughness R_{max} = 0.05 m. The minimum diameter of a landing site was set to r_{min} = 5 m. The evaluation begins at t = 3086.2$\overline{3}$ s = T_{1000} where the lander is about one kilometer above ground. At time T_{254} = t = 3120.9$\overline{3}$ s the lander is about 254 meters above the ground where the LIDAR reaches the necessary ground resolution.

The question is whether the defined parameters δ_{max}, R_{max} and r_{min} are well-chosen for finding sufficient landing sites in the given scenario 6. This is answered by Fig. 3.39.

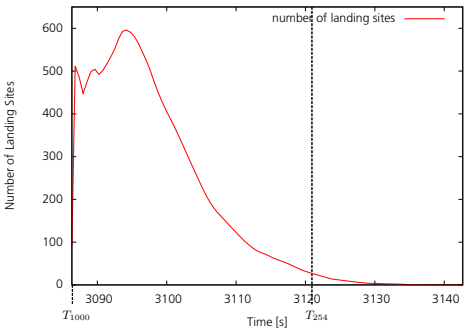


Figure 3.39: Number of found landing sites from about one kilometer above lunar surface up to touchdown

Specifically, it was necessary to clarify the question of whether landing sites at the time T_{254} could be identified. Fig. 3.39 shows that at the beginning of the processing, only few landing sites are identified, which is due to the fact that the entire data scope of a LIDAR data set is not yet available. In higher altitudes, up to 600 landing sites are identified. This number decreases to ≈24 candidates at the time T_{254}. After time T_{254} landing sites can be identified for a further 15 s.

This result cannot provide information about the quality of the detected landing sites. But it shows that a sufficient number of landing sites can be

found to make a selection if an originally intended landing site is no longer available.

3.6 Visual SLAM

The described Crater Navigation module (Sect. 3.1), the 3D-Matching Processing Chain modules (Sect. 3.3), and the Binary Shadow Matching module (Sect. 3.4) all provide an absolute position. The visual SLAM module on the other hand provides relative position updates. See Fig. 3.40 for an overview where visual SLAM is used as a relative localization method.

Simultaneous Localization and Mapping (SLAM) generally consists of a *localization* step where a sensor pose (and, with that, a vehicle pose) is estimated by the registration between current sensor data (here, the tracked features from Sect. 3.2) and an environment map, and a *mapping* step where this map is generated and updated based on the sensor data. Since localization and mapping highly depend on each other, it marks a typical chicken-and-egg problem and thus requires an adequate initialization of the pose or the map. Within the visual SLAM module, 2D and 3D features from arbitrary sensor sources can be processed. A complete description of the visual SLAM module is published in [4], which describes all the necessary math and provides visual path estimation results from unmanned aircraft data examples.

The *mapping* algorithm generates a sparse but globally consistent 3D map of the visible features. It can be used for external purposes (e.g. obstacle or landing site detection), but will be used for localization only in the context of project ATON. This is done by combining multi-view monocular triangulation (i.e. from 2D features) and a 3D coordinate data fusion where 3D features are available. Triangulation always requires the extrinsic camera orientations for all used images. They are derived from the updated vehicle states which are interpolated to the particular image time stamps. In contrast to sole monocular triangulation which suffers from scale uncertainty and drift, a data fusion with 3D features allows the correct map scaling to be maintained over time. However, monocular mapping is still the main source of map objects since 3D information is only available for a small subset of tracked features.

The *localization* algorithm estimates a vehicle pose with non-linear optimization. It is basically a monocular camera resectioning from correspond-

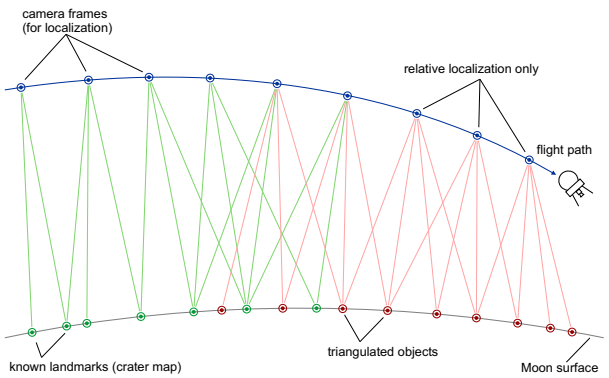


Figure 3.40: Camera localization from known landmarks and a priori unknown objects triangulated during the flight. A subset of the camera images (key frames) is used to map these

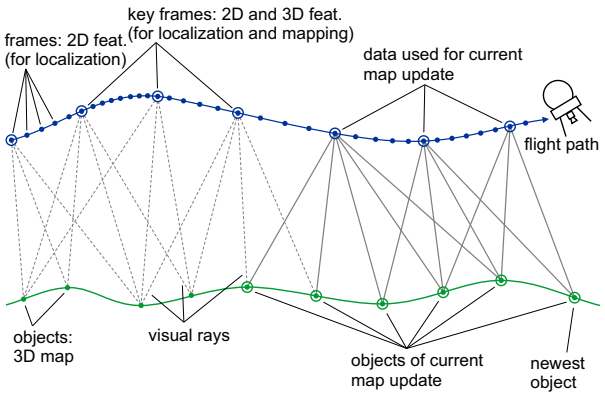


Figure 3.41: Visual SLAM principle, with 3D key frame mapping and higher-frequent monocular localization

ing 2D features and geodetic 3D map objects. Initialized by the inertially predicted vehicle state, the reprojection errors of the visible features are to be minimized, and this minimization returns an updated vehicle pose where the features fit to the map objects. Updates are generated with the highest possible frequency, i.e. for every new camera frame if possible. In contrast to that, mapping is done only for key frames, i.e. for frames with salient key features where it is expected that triangulation will return suitable 3D map object coordinates or where a retriangulation with a higher number of input features is beneficial. Fig. 3.41 illustrates this principle. To be independent of frame rates and flight altitudes, a feature is chosen as a key feature if the angle between the projection rays of this and a previous key feature exceeds a certain threshold (e.g. 3 deg is a good value). Newly identified features are always key features. Key frame mapping reduces computational load while the map quality should not be reduced that much. It is obvious that the relative number of key features is increased during fast and low flights where the optical flow is high.

Next to localization and mapping which generally consists of camera resectioning and object triangulation, the visual SLAM module includes some further filtering to increase the robustness. Implemented components are outlier filtering, i.e. removing the features and objects with significantly high residuals within the optimization, and update filtering, i.e. removing localization results if they are definitely not plausible when compared with the prediction.

In the ATON project the visual SLAM module is part of the Navigation Filter module (Sect. 3.7). Fig. 3.42 provides an overview of the interaction between the SLAM module and the central state estimator.

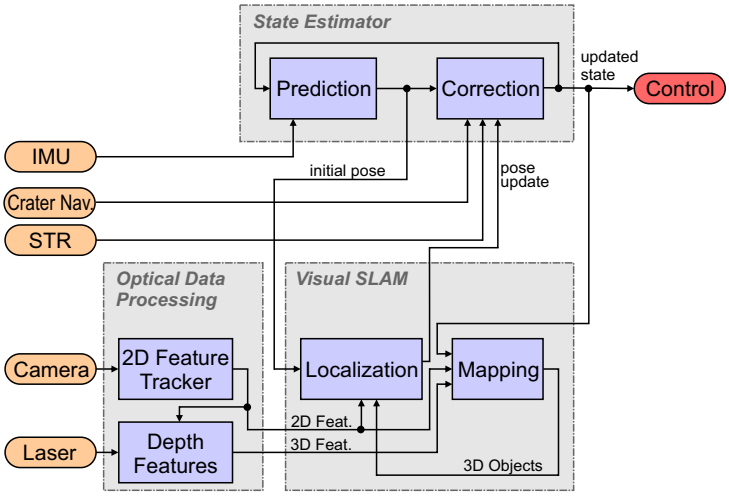


Figure 3.42: The optical navigation architecture with three separate modules

3.7 Navigation Filter

In this section, the *Navigation Filter* module is described, which has been published in [1]. This section is organized as follows: first, the state definitions in the total state space and the error state space will be given. Subsequently, the software architecture of the navigation system will be described. This is followed by the description of the strapdown navigation algorithm, continuing with the sensor data fusion algorithm and the Unscented Kalman Filter (UKF). Finally, this section closes with the description of the process and measurement models of the UKF.

3.7.1 State Definition

The output of the navigation system consists of three distinct parts: the navigation solution or total state $\boldsymbol{x}$, the error state $\delta\boldsymbol{x}$, and the covariance of the error state $\boldsymbol{P}$.

3.7.2 State

This is the most important part of the output of the navigation system since the total state $\boldsymbol{x}$ is the navigation solution. It consists of the position $\boldsymbol{r}_{\mathrm{M}}$, the velocity $\boldsymbol{v}_{\mathrm{M}}$, and the attitude $\boldsymbol{q}_{\mathrm{B}}^{\mathrm{M}}$, see (3.18). The update of this output is triggered with every new inertial measurement of the IMU. Thus, it is updated at a high rate, such as 100 Hz.

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{r}_{\mathrm{M}} \\ \boldsymbol{v}_{\mathrm{M}} \\ \boldsymbol{q}_{\mathrm{B}}^{\mathrm{M}} \end{bmatrix} \in \mathbb{R}^{10} \tag{3.18}$$

3.7.3 Error State

Since the resulting navigation solution or total state $\boldsymbol{x}$ accumulates errors due to noisy and biased inertial measurements and model uncertainties, it has to be corrected to remain useful. Therefore, the error state $\delta\boldsymbol{x}$ is defined as shown in (3.19). It consists of the position error $\delta\boldsymbol{r}_{\mathrm{M}}$, the velocity

error $\delta \boldsymbol{v}_{\mathrm{M}}$, and the attitude error $\delta \boldsymbol{\theta}_{\mathrm{M}}^{\mathrm{M}}$[2] to correct the total state $\boldsymbol{x}$. Additionally, sensor errors of the inertial measurement unit like bias errors of the accelerometer ($\boldsymbol{b}_{\boldsymbol{a}_{\mathrm{B}}}$) and the gyroscope ($\boldsymbol{b}_{\boldsymbol{\omega}_{\mathrm{B}}}$) are included in the error state. Finally, we include the deviation of the modeled and the true gravitation vector $\boldsymbol{b}_{\boldsymbol{g}_{\mathrm{M}}}$ in the error state. $\delta \boldsymbol{x}$ is continuously updated via the UKF by fusing all available measurements, in such a way that the resulting variance of the error state is minimal. These updates are performed at a much lower rate than the propagation of the total state $\boldsymbol{x}$.

$$\delta \boldsymbol{x} = \begin{bmatrix} \delta \boldsymbol{r}_{\mathrm{M}} \\ \delta \boldsymbol{v}_{\mathrm{M}} \\ \delta \boldsymbol{\theta}_{\mathrm{M}}^{\mathrm{M}} \\ \boldsymbol{b}_{\boldsymbol{a}_{\mathrm{B}}} \\ \boldsymbol{b}_{\boldsymbol{\omega}_{\mathrm{B}}} \\ \boldsymbol{b}_{\boldsymbol{g}_{\mathrm{M}}} \end{bmatrix} \in \mathbb{R}^{18} \tag{3.19}$$

The Kalman Filter equations and the Unscented transformation used in the UKF assume the elements of the (error) state vector to be uncorrelated. Since a rotation quaternion $\boldsymbol{q}$ has the additional condition $\|\boldsymbol{q}\| = 1$, a different representation is needed to express the attitude error $\delta \boldsymbol{q}_{\mathrm{M}}^{\mathrm{M}}$ in the error state $\delta \boldsymbol{x}$.

Therefore, the attitude error is expressed as an axis–angle vector $\delta \boldsymbol{\theta}_{\mathrm{M}}^{\mathrm{M}}$:

$$\delta \boldsymbol{\theta}_{\mathrm{M}}^{\mathrm{M}} = \boldsymbol{e} \cdot \theta, \qquad \boldsymbol{e} \in \mathbb{R}^3,\ \boldsymbol{e}^{\mathrm{T}} \cdot \boldsymbol{e} = 1 \tag{3.20}$$

Where $\boldsymbol{e}$ being the axis of the rotation and $\theta \geq 0$ with $\theta \in \mathbb{R}$ is the rotation angle. This representation consists of three uncorrelated elements.

The axis–angle representation of the attitude error $\delta \boldsymbol{\theta}_{\mathrm{M}}^{\mathrm{M}}$ can be transformed to a rotation quaternion representation $\delta \boldsymbol{q}_{\mathrm{M}}^{\mathrm{M}}$ using the following formula:

$$\delta \boldsymbol{q}_{\mathrm{M}}^{\mathrm{M}} = \begin{bmatrix} \boldsymbol{e} \cdot \sin\left(\frac{\theta}{2}\right) \\ \cos\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{3.21}$$

Using the two state definitions, the true navigation solution $\boldsymbol{x}$ can be expressed using the estimated navigation solution $\tilde{\boldsymbol{x}}$ and the error state $\delta \boldsymbol{x}$. This is defined as follows:

$$\boldsymbol{x} := \begin{bmatrix} \tilde{\boldsymbol{r}}_{\mathrm{M}} + \delta \boldsymbol{r}_{\mathrm{M}} \\ \tilde{\boldsymbol{v}}_{\mathrm{M}} + \delta \boldsymbol{v}_{\mathrm{M}} \\ \delta \boldsymbol{q}_{\mathrm{M}}^{\mathrm{M}} \otimes \tilde{\boldsymbol{q}}_{\mathrm{B}}^{\mathrm{M}} \end{bmatrix} \tag{3.22}$$

3.7.4 Covariance

The third output is the covariance matrix $\boldsymbol{P}$ of the error state $\delta \boldsymbol{x}$. This covariance matrix represents the uncertainty of the estimated error state and therefore also represents the uncertainty of the navigation solution.

$$\boldsymbol{P} \in \mathbb{R}^{18 \times 18} \tag{3.23}$$

[2] The attitude error $\delta \boldsymbol{\theta}_{\mathrm{M}}^{\mathrm{M}}$ is given as a slight tilt of the computation frame M.

3.7.5 Software Architecture

Following the description of the outputs of the navigation system, we now provide a detailed description of its software architecture. In Fig. 3.43, a visual description of the complete system can be found. The figure shows the system as a block diagram with two types of inputs and the three output types as described above.

The inputs are divided into measurements from the inertial measurement unit such as acceleration $\tilde{\boldsymbol{a}}_B$ and angular rate $\tilde{\boldsymbol{\omega}}_B$ which are provided at a high rate, and other sensor measurements $\boldsymbol{h}$ with associated standard deviation $\boldsymbol{\sigma}_{\boldsymbol{h}}$ like absolute position, velocity or attitude measurements. These measurements are provided at a lower rate.

The outputs of the navigation system as depicted in the block diagram are the navigation solution or total state $\boldsymbol{x}$, the error state $\delta\boldsymbol{x}$, and the covariance of the error state $\boldsymbol{P}$. These output types have been described in detail in the *State Definition* part of this section.

The calculations performed by the navigation system are divided into different modules. These modules are triggered by their inputs and therefore run at different rates. In Fig. 3.43 these different modules with their associated rates are distinguished by color. The modules and their associated data flow marked orange are executed at 100 Hz. The other modules and data flow paths marked blue are executed at a lower rate, such as 1 Hz. This architecture allows the navigation system to follow the high frequency motion of the spacecraft very accurately on the basis of the high rate inertial measurements, while the error state follows a lower dynamic [74].

The module *Error Compensation* is triggered by each new inertial sensor measurement. In this module the incoming raw inertial sensor measurements ($\tilde{\boldsymbol{a}}_B$ and $\tilde{\boldsymbol{\omega}}_B$) are corrected using the estimated sensor errors ($\boldsymbol{b}_{\boldsymbol{a}_B}$ and $\boldsymbol{b}_{\boldsymbol{\omega}_B}$) included in the error state. With that, the corrected inertial sensor measurements are defined as:

$$\boldsymbol{a}_B = \tilde{\boldsymbol{a}}_B + \boldsymbol{b}_{\boldsymbol{a}_B} \tag{3.24}$$

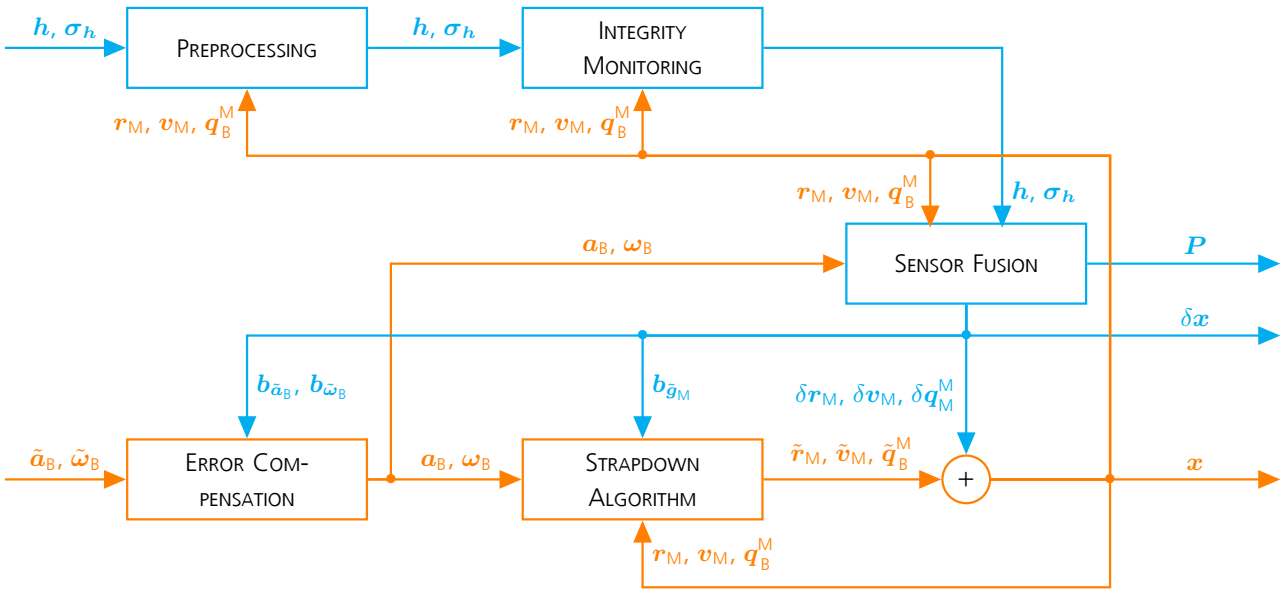


Figure 3.43: Block diagram of the software architecture of the navigation system. The modules with the corresponding data flow are shown. Different update rates are indicated by color

$$\boldsymbol{\omega}_{\mathrm{B}} = \tilde{\boldsymbol{\omega}}_{\mathrm{B}} + \boldsymbol{b}_{\boldsymbol{\omega}_{\mathrm{B}}} \tag{3.25}$$

These corrected inertial sensor measurements are used as an input for the *Strapdown Algorithm* module. This module uses inertial navigation to calculate the next navigation solution $\tilde{\boldsymbol{x}}(t + \Delta t)$ based on the current navigation solution $\boldsymbol{x}(t)$ and the current (corrected) inertial measurements $\boldsymbol{a}_{\mathrm{B}}$ and $\boldsymbol{\omega}_{\mathrm{B}}$, where Δt is the time between two inertial measurements. Additionally, this module uses the bias error $\boldsymbol{b}_{\tilde{\boldsymbol{g}}_{\mathrm{M}}}$ to correct uncertainties of the gravitational model. The predicted total state $\tilde{\boldsymbol{x}}$ consists of the predicted position $\tilde{\boldsymbol{r}}_{\mathrm{M}}$, the predicted velocity $\tilde{\boldsymbol{v}}_{\mathrm{M}}$, and the predicted attitude $\tilde{\boldsymbol{q}}_{\mathrm{B}}^{\mathrm{M}}$. This module is described in more detail in the *Strapdown Inertial Navigation Equations* part of this section.

The module marked with the addition symbol uses the current predicted total state $\tilde{\boldsymbol{x}}$ and the latest estimated errors for the position $\delta\boldsymbol{r}_{\mathrm{M}}$, the velocity $\delta\boldsymbol{v}_{\mathrm{M}}$, and the attitude $\delta\boldsymbol{q}_{\mathrm{M}}^{\mathrm{M}}$ which are contained in the error state $\delta\boldsymbol{x}$ to calculate the corrected total state $\boldsymbol{x}$.

The module *Preprocessing* is triggered by the other sensor measurements $\boldsymbol{h}$ and therefore runs at a lower rate. This module is used to perform various tasks depending on the type of measurement. For absolute position sensor measurements, a typical task is the compensation of the lever arm between the sensor or antenna and the body frame. To perform such tasks the navigation solution ($\boldsymbol{r}_{\mathrm{M}}$, $\boldsymbol{v}_{\mathrm{M}}$ and $\boldsymbol{q}_{\mathrm{B}}^{\mathrm{M}}$) is also used as an input to provide the required data.

The module *Integrity Monitoring* is used to prevent invalid sensor data to be incorporated in the calculations of the navigation system. As long as the IMU is working as specified, inconsistencies in the sensor fusion may only occur due to invalid sensor inputs of the aiding sensors. These inconsistencies can cause instability in the fusion and therefore cause faulty navigation solutions. Therefore, the range of the sensor measurements is checked against specified bounds. Furthermore, the outputs of the navigation system (e.g. the total state $\boldsymbol{x}$ and the covariance of the error state $\boldsymbol{P}$) can be used to assess the input data statistically. For that, the difference of the measurements $\boldsymbol{z}$ and the predicted measurements (also know as the *innovation* $\boldsymbol{y}$) can be assessed versus the assumed uncertainty given by the residual covariance $\boldsymbol{S}$. This tests the so-called Normalized Innovations Squared (NIS), which is closely related to the Chi-Square $\left(\chi^2\right)$ tests. See [59] and [8] for more details. The outcome of these tests determines if the measurement data should be rejected or forwarded to the *Sensor Fusion* module.

In the module *Sensor Fusion*, the preprocessed sensor data $\boldsymbol{h}$ with associated standard deviation $\boldsymbol{\sigma}_{\boldsymbol{h}}$, the inertial measurements $\boldsymbol{a}_{\mathrm{B}}$, $\boldsymbol{\omega}_{\mathrm{B}}$, and the total state $\boldsymbol{x}$ are fused to estimate the new error state estimate $\delta\boldsymbol{x}$ with the corresponding covariance matrix $\boldsymbol{P}$. This module is described in detail in the *Sensor Fusion* part of this section.

3.7.6 Strapdown Inertial Navigation Equations

In this section, the equations used for the inertial navigation in the *Strapdown Algorithm* module are described. As described earlier, the objective of the *Strapdown Algorithm* module is the calculation of the next navigation solution $\tilde{\boldsymbol{x}}(t + \Delta t)$ based on the current navigation solution $\boldsymbol{x}(t)$, the current (corrected) inertial measurements $\boldsymbol{a}_{\mathrm{B}}$ and $\boldsymbol{\omega}_{\mathrm{B}}$, and the bias error $\boldsymbol{b}_{\tilde{\boldsymbol{g}}_{\mathrm{M}}}$ of the modeled gravitation vector $\tilde{\boldsymbol{g}}_{\mathrm{M}}$.

The modeled gravitation vector $\tilde{\boldsymbol{g}}_{\mathrm{M}}$ is the gravitational acceleration above the lunar surface. It depends on the current position $\boldsymbol{r}_{\mathrm{M}}$ and can be expressed using a combination of Newton's law of gravitation and the centrifugal acceleration:

$$\boldsymbol{\Upsilon}_{\mathrm{M}}(\boldsymbol{r}_{\mathrm{M}}) = \tilde{\boldsymbol{g}}_{\mathrm{M}} \tag{3.26}$$

$$= -\boldsymbol{r}_{\mathrm{M}} \cdot \frac{\mu}{\|\boldsymbol{r}_{\mathrm{M}}\|^3} - \boldsymbol{\omega}_{\mathrm{M}} \times (\omega_{\mathrm{M}} \times \boldsymbol{r}_{\mathrm{M}}) \tag{3.27}$$

where $\mu = G \cdot M$ is the standard gravitational parameter of the Moon, with the gravitational constant $G = 6.674 \cdot 10^{-11}\,\mathrm{Nm^2kg^{-2}}$, the Moon's mass $M = 7.353 \cdot 10^{22}$ kg [28], and $\boldsymbol{\omega}_{\mathrm{M}}$ is the angular rate of the celestial body's rotation. This modeled term only approximates the highly irregular lunar gravity field and its modelling errors are corrected using an estimated bias error $\boldsymbol{b}_{\tilde{\boldsymbol{g}}_{\mathrm{M}}}$ from the Sensor Fusion module.

For the calculation of the next navigation solution $\tilde{\boldsymbol{x}}(t + \Delta t)$ the state's time derivative $\dot{\boldsymbol{x}}$ can be expressed as a kinematic system of differential equations:

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} \dot{\boldsymbol{r}}_{\mathrm{M}} \\ \dot{\boldsymbol{v}}_{\mathrm{M}} \\ \dot{\boldsymbol{q}}_{\mathrm{B}}^{\mathrm{M}} \end{bmatrix} \tag{3.28}$$

$$= f\left(\boldsymbol{x}(t), \left[\boldsymbol{a}_{\mathrm{B}}, \boldsymbol{\omega}_{\mathrm{B}}, \boldsymbol{b}_{\tilde{\boldsymbol{g}}_{\mathrm{M}}}, \boldsymbol{\omega}_{\mathrm{M}}\right](t)\right) \tag{3.29}$$

$$= \begin{bmatrix} \boldsymbol{v}_{\mathrm{M}} \\ \boldsymbol{R}_{\mathrm{B}}^{\mathrm{M}} \boldsymbol{a}_{\mathrm{B}} + \boldsymbol{\Upsilon}_{\mathrm{M}}(\boldsymbol{r}_{\mathrm{M}}) + \boldsymbol{b}_{\tilde{\boldsymbol{g}}_{\mathrm{M}}} - 2\boldsymbol{\omega}_{\mathrm{M}} \times \boldsymbol{v}_{\mathrm{M}} \\ \boldsymbol{R}_{\mathrm{B}}^{\mathrm{M}} \cdot \boldsymbol{\Omega}_{\mathrm{B}} - \boldsymbol{\Omega}_{\mathrm{M}} \cdot \boldsymbol{R}_{\mathrm{B}}^{\mathrm{M}} \end{bmatrix} \tag{3.30}$$

where $\boldsymbol{R}_{\mathrm{B}}^{\mathrm{M}} \in \mathbb{R}^{3\times3}$ is the rotation matrix representation of the rotation quaternion $\boldsymbol{q}_{\mathrm{B}}^{\mathrm{M}}$, and $\boldsymbol{\Omega}$ is the representation of an angular rate vector $\boldsymbol{\omega}$ as a skew-symmetric or anti-symmetric matrix:

$$\boldsymbol{\Omega} = \begin{bmatrix} 0 & -\omega_z & +\omega_y \\ +\omega_z & 0 & -\omega_x \\ -\omega_y & +\omega_x & 0 \end{bmatrix} \in \mathbb{R}^{3\times3} \tag{3.31}$$

The other terms in (3.30) are the Coriolis acceleration $2 \cdot \boldsymbol{\omega}_{\mathrm{M}} \times \boldsymbol{v}_{\mathrm{M}}$ and the change in attitude $\boldsymbol{\Omega}_{\mathrm{M}} \cdot \boldsymbol{R}_{\mathrm{B}}^{\mathrm{M}}$ resulting from the angular rate of the celestial body's rotation $\boldsymbol{\omega}_{\mathrm{M}}$.

With that, the navigation solution for the next time step $t + \Delta t$ can be determined, with Δt being the time between two inertial measurements. Using the Euler method the navigation solution $\tilde{\boldsymbol{x}}(t + \Delta t)$ can be easily calculated:

$$\boldsymbol{x}(t + \Delta t) = \boldsymbol{x}(t) + \Delta t \cdot \dot{\boldsymbol{x}}(t) \tag{3.32}$$

3.7.7 Sensor Fusion

The *Sensor Fusion* module is the main part of this work. As mentioned in the previous section, it fuses the preprocessed sensor data $\boldsymbol{h}$ with associated standard deviation $\boldsymbol{\sigma}_{\boldsymbol{h}}$, the inertial measurements $\boldsymbol{a}_{\mathrm{B}}$ and $\boldsymbol{\omega}_{\mathrm{B}}$, and the total state $\boldsymbol{x}$ to estimate the sensor errors and the error contained in the navigation solution. These errors are combined in the error state $\delta\boldsymbol{x}$ as shown in (3.19).

3.7.8 Unscented Kalman Filter

For the fusion algorithm the UKF has been chosen.

The UKF has some advantages over the widely used Extended Kalman Filter (EKF). The EKF uses the non-linear state transition and observation models for the state estimation but uses linearization to be able to apply the standard Kalman Filter equations for the covariance calculation. The UKF, on the other hand, uses a deterministic sampling technique to calculate the state and covariance using the non-linear models directly. Therefore, the UKF is able to estimate the state of the non-linear system more accurately than the EKF [42]. Furthermore the EKF is difficult to implement and to maintain [40] because the Jacobian and Hessian matrices of the process and observation models have to be derived, which is not necessary for the UKF.

In the following the implemented UKF algorithm is explained step-wise. The implementation is based on different papers by Julier, Uhlmann, Wan and Wu [41], [42], [90], [40] and [92].

As mentioned in the *State Definition* and *Architecture* parts of this section the UKF is used to update the error state, not the total state directly. Therefore, the description of the implemented UKF is given in the error state space.

Since the estimated state from the previous time step along with the current measurement are needed in the prediction and update steps of the UKF to predict the current state, there has to be a distinction between the states corresponding to different time stamps. In the following, the notation $\delta\tilde{\boldsymbol{x}}_{n|m}$ represents the estimate of $\delta\boldsymbol{x}$ at time n given measurements up to and including time $m \leq n$.

The following nine steps are executed for each time step to first predict the error state $\delta\tilde{\boldsymbol{x}}_{k|k}$ at time k to the next time step $k+1$ and to update the error state $\delta\tilde{\boldsymbol{x}}_{k+1|k}$ with the current measurements $\boldsymbol{z}_{k+1}$ to obtain the corrected error state $\delta\tilde{\boldsymbol{x}}_{k+1|k+1}$:

1. The estimated error state $\delta\tilde{\boldsymbol{x}}_{k|k}$ and covariance $\boldsymbol{P}_{k|k}$ of the current time step k are augmented with the mean and covariance of the process noise $\boldsymbol{w}_{k+1}$ and measurement noise $\boldsymbol{v}_{k+1}$ of the next time step $k+1$. The measurement noise and its covariance depend on the actual measurements available at time step k. Therefore, the dimension of $\boldsymbol{v}_{k+1}$ and $\boldsymbol{R}_{k+1}$ varies over time.

$$\delta\tilde{\boldsymbol{x}}^a_{k|k} = \begin{bmatrix} \delta\tilde{\boldsymbol{x}}^{\mathrm{T}}_{k|k} & E[\boldsymbol{w}^{\mathrm{T}}_{k+1}] & E[\boldsymbol{v}^{\mathrm{T}}_{k+1}] \end{bmatrix}^{\mathrm{T}} \tag{3.33}$$

$$\boldsymbol{P}^a_{k|k} = \begin{bmatrix} \boldsymbol{P}_{k|k} & 0 & 0 \\ 0 & \boldsymbol{Q}_{k+1} & 0 \\ 0 & 0 & \boldsymbol{R}_{k+1} \end{bmatrix} \tag{3.34}$$

2. A set of $2L+1$ sigma points is derived from the augmented state $\delta\tilde{\boldsymbol{x}}^a_{k|k}$ and covariance $\boldsymbol{P}^a_{k|k}$, where $L = 2{\cdot}\dim(\boldsymbol{x}){+}\dim(\boldsymbol{z}_{k+1})$ is the dimension of the augmented state. With $\left[\sqrt{(L+\lambda)\boldsymbol{P}^a_{k|k}}\right]_i$ is the ith column of the matrix square root[3] of $(L{+}\lambda)\boldsymbol{P}^a_{k|k}$ the sigma points $\boldsymbol{\chi}^i_{k|k}$, with $i \in [0, 2L]$ are defined as follows:

$$\boldsymbol{\chi}^0_{k|k} = \delta\tilde{\boldsymbol{x}}^a_{k|k} \tag{3.35}$$

[3] The matrix square root $\boldsymbol{A}$ of matrix $\boldsymbol{B}$ satisfies $\boldsymbol{B} \triangleq \boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}$. This could be calculated using the numerically efficient and stable Cholesky decomposition.

$$\boldsymbol{\chi}^{i}_{k|k} = \delta\tilde{\boldsymbol{x}}^{a}_{k|k} + \left[\sqrt{(L+\lambda)\boldsymbol{P}^{a}_{k|k}}\right]_{i}, \qquad i \in [1, L] \tag{3.36}$$

$$\boldsymbol{\chi}^{i+L}_{k|k} = \delta\tilde{\boldsymbol{x}}^{a}_{k|k} - \left[\sqrt{(L+\lambda)\boldsymbol{P}^{a}_{k|k}}\right]_{i}, \qquad i \in [1, L] \tag{3.37}$$

3. The sigma points $\boldsymbol{\chi}^{i}_{k|k}$ are propagated through the transition function $f : \mathbb{R}^{L} \to \mathbb{R}^{\dim(\delta\boldsymbol{x})+\dim(\boldsymbol{z}_{k+1})}$. This function is described in the *Process Model* part of this section.

$$\boldsymbol{\chi}^{i}_{k+1|k} = f\left(\boldsymbol{\chi}^{i}_{k|k}\right), \quad i \in [0, 2L] \tag{3.38}$$

4. The transformed sigma points $\boldsymbol{\chi}^{i}_{k+1|k}$ are weighted and recombined to calculate the predicted state $\delta\tilde{\boldsymbol{x}}_{k+1|k}$ and covariance $\boldsymbol{P}_{k+1|k}$.

$$\delta\tilde{\boldsymbol{x}}_{k+1|k} = \sum_{i=0}^{2L} W^{i}_{s}\boldsymbol{\chi}^{i}_{k+1|k} \tag{3.39}$$

$$\boldsymbol{P}_{k+1|k} = \sum_{i=0}^{2L} W^{i}_{c}\left[\boldsymbol{\chi}^{i}_{k+1|k} - \delta\tilde{\boldsymbol{x}}_{k+1|k}\right]\left[\boldsymbol{\chi}^{i}_{k+1|k} - \delta\tilde{\boldsymbol{x}}_{k+1|k}\right]^{\mathrm{T}} \tag{3.40}$$

Where the weights are given by:

$$W^{0}_{s} = \frac{\lambda}{L+\lambda} \tag{3.41}$$

$$W^{0}_{c} = \frac{\lambda}{L+\lambda} + (1 - \alpha^{2} + \beta) \tag{3.42}$$

$$W^{i}_{s} = W^{i}_{c} = \frac{1}{2(L+\lambda)} \tag{3.43}$$

with $\lambda = \alpha^{2}(L+\kappa) - L$.
The parameters α and κ control the spread of the sigma points, whereas β is related to the distribution of $\delta\boldsymbol{x}$. Normal values are $\alpha = 10^{-3}$, $\kappa = 0$ and $\beta = 2$. If the true distribution of $\delta\boldsymbol{x}$ is Gaussian, $\beta = 2$ is optimal [90].

5. For the update of the predicted state $\delta\tilde{\boldsymbol{x}}_{k+1|k}$, the transformed sigma points $\boldsymbol{\chi}^{i}_{k+1|k}$ are projected through the observation function $h : \mathbb{R}^{\dim(\delta\boldsymbol{x})+\dim(\boldsymbol{z}_{k+1})} \to \mathbb{R}^{\dim(\boldsymbol{z}_{k+1})}$ to calculate the gamma points $\boldsymbol{\gamma}^{i}_{k+1}$. This function is described in the *Measurement Model* part of this section.

$$\boldsymbol{\gamma}^{i}_{k+1} = h\left(\boldsymbol{\chi}^{i}_{k+1|k}\right) \quad i \in [0, 2L] \tag{3.44}$$

6. To calculate the predicted measurement $\tilde{\boldsymbol{z}}_{k+1}$ and the predicted measurement covariance $\boldsymbol{P}_{z_{k+1}z_{k+1}}$, the gamma points $\boldsymbol{\gamma}^{i}_{k+1}$ are recombined.

$$\tilde{\boldsymbol{z}}_{k+1} = \sum_{i=0}^{2L} W^{i}_{s}\boldsymbol{\gamma}^{i}_{k+1} \tag{3.45}$$

$$\boldsymbol{P}_{z_{k+1}z_{k+1}} = \sum_{i=0}^{2L} W^{i}_{c}\left[\boldsymbol{\gamma}^{i}_{k+1} - \tilde{\boldsymbol{z}}_{k+1}\right]\left[\boldsymbol{\gamma}^{i}_{k+1} - \tilde{\boldsymbol{z}}_{k+1}\right]^{\mathrm{T}} \tag{3.46}$$

7. For the calculation of the state-measurement cross covariance matrix $\boldsymbol{P}_{x_{k+1}z_{k+1}}$, the transformed sigma points $\boldsymbol{\chi}^{i}_{k+1|k}$ and the gamma points $\boldsymbol{\gamma}^{i}_{k+1}$ are recombined.

$$\boldsymbol{P}_{x_{k+1}z_{k+1}} = \sum_{i=0}^{2L} W^{i}_{c}\left[\boldsymbol{\chi}^{i}_{k+1|k} - \delta\tilde{\boldsymbol{x}}_{k+1|k}\right]\left[\boldsymbol{\gamma}^{i}_{k+1} - \tilde{\boldsymbol{z}}_{k+1}\right]^{\mathrm{T}} \tag{3.47}$$

8. Using the predicted measurement covariance matrix $\boldsymbol{P}_{z_{k+1}z_{k+1}}$ and the state-measurement cross covariance matrix $\boldsymbol{P}_{x_{k+1}z_{k+1}}$, the Kalman gain matrix K_{k+1} is computed.

$$K_{k+1} = \boldsymbol{P}_{x_{k+1}z_{k+1}}\boldsymbol{P}^{-1}_{z_{k+1}z_{k+1}} \tag{3.48}$$

9. With the calculated Kalman gain matrix K_{k+1} and the predicted measurement vector $\tilde{\boldsymbol{z}}_{k+1}$, the predicted state $\delta\tilde{\boldsymbol{x}}_{k+1|k}$ and covariance $\boldsymbol{P}_{k+1|k}$ can be updated in the following way.

$$\delta\hat{\boldsymbol{x}}_{k+1|k+1} = \delta\tilde{\boldsymbol{x}}_{k+1|k} + K_{k+1}(\boldsymbol{z}_{k+1} - \tilde{\boldsymbol{z}}_{k+1}) \tag{3.49}$$

$$\boldsymbol{P}_{k+1|k+1} = \boldsymbol{P}_{k+1|k} - K_{k+1}\boldsymbol{P}_{z_{k+1}z_{k+1}}K_{k+1}^{\mathrm{T}} \tag{3.50}$$

3.7.9 Process Model

In the description of the UKF the transition function $f : \mathbb{R}^L \rightarrow \mathbb{R}^{\dim(\delta\boldsymbol{x})+\dim(\boldsymbol{z}_{k+1})}$ has been mentioned. This function represents the transition of the error state in time stamp k to the new error state in time stamp $k+1$.

The sigma points have been constructed in a way that each point can be divided into three parts. The first part corresponds with the actual error state vector $\delta\boldsymbol{x}$, the second part corresponds with the process noise $\boldsymbol{w}_{k+1}$ and the third part corresponds to the measurement noise $\boldsymbol{v}_{k+1}$. Each part has to be handled individually as can be seen in (3.51) to (3.53). The state vector part is projected through the actual transition function $f' : \mathbb{R}^{\dim(\delta\boldsymbol{x})} \rightarrow \mathbb{R}^{\dim(\delta\boldsymbol{x})}$, whereas the process noise part is added to the result. The part corresponding to the measurement noise is of importance when projecting the transformed sigma points through the measurement model. Therefore this part remains untouched by the transition function.

$$\boldsymbol{\chi}_{k+1|k} = f\left(\boldsymbol{\chi}_{k|k}\right) \tag{3.51}$$

$$= f\left(\begin{bmatrix}\delta\boldsymbol{x}\\ \boldsymbol{w}_{k+1}\\ \boldsymbol{v}_{k+1}\end{bmatrix}\right) \tag{3.52}$$

$$= \begin{bmatrix}f'(\delta\boldsymbol{x}) + \boldsymbol{w}_{k+1}\\ \boldsymbol{v}_{k+1}\end{bmatrix} \tag{3.53}$$

Since the sensor fusion uses the UKF in updating the error state $\delta\boldsymbol{x}$, the transition function f' has to be formulated in the error state space. Based on the kinematic system of differential equations (3.30) from the *Strapdown Inertial Navigation Equations* section, the differential equations of the navigation error can be derived using the relationship described in (3.22). The resulting equations can be found in (3.54) to (3.56). From (3.56) it also follows that the definition of the attitude error $\delta\boldsymbol{\theta}_{\mathrm{M}}^{\mathrm{M}}$ in the M frame has lower error dynamics than the definition in the B frame.

$$\delta\dot{\boldsymbol{r}}_{\mathrm{M}} = \delta\boldsymbol{v}_{\mathrm{M}} \tag{3.54}$$

$$\delta\dot{\boldsymbol{v}}_{\mathrm{M}} = \left(\delta\boldsymbol{R}_{\mathrm{M}}^{\mathrm{M}} - \boldsymbol{I}\right)\cdot\tilde{\boldsymbol{R}}_{\mathrm{B}}^{\mathrm{M}}\cdot\tilde{\boldsymbol{a}}_{\mathrm{B}} + \delta\boldsymbol{R}_{\mathrm{M}}^{\mathrm{M}}\cdot\tilde{\boldsymbol{R}}_{\mathrm{B}}^{\mathrm{M}}\cdot\boldsymbol{b}_{\boldsymbol{a}_{\mathrm{B}}} + \boldsymbol{\Upsilon}_{\mathrm{M}}\left(\tilde{\boldsymbol{r}}_{\mathrm{M}} + \delta\boldsymbol{r}_{\mathrm{M}}\right) - \boldsymbol{\Upsilon}_{\mathrm{M}}\left(\tilde{\boldsymbol{r}}_{\mathrm{M}}\right) + \boldsymbol{b}_{\tilde{\boldsymbol{g}}_{\mathrm{M}}} - 2\cdot\boldsymbol{\omega}_{\mathrm{M}}\times\delta\boldsymbol{v}_{\mathrm{M}} \tag{3.55}$$

$$\delta\dot{\boldsymbol{R}}_{\mathrm{M}}^{\mathrm{M}} = \delta\boldsymbol{R}_{\mathrm{M}}^{\mathrm{M}}\cdot\left(\tilde{\boldsymbol{R}}_{\mathrm{B}}^{\mathrm{M}}\cdot\boldsymbol{\Omega}_{\boldsymbol{b}_{\boldsymbol{\omega}_{\mathrm{B}}}}\cdot\tilde{\boldsymbol{R}}_{\mathrm{B}}^{\mathrm{M}^{\mathrm{T}}} + \boldsymbol{\Omega}_{\mathrm{M}}\right) - \boldsymbol{\Omega}_{\mathrm{M}}\cdot\delta\boldsymbol{R}_{\mathrm{M}}^{\mathrm{M}} \tag{3.56}$$

With these differential equations and a defined time increment Δt, the function f' can be defined as follows:

$$f'(\delta\boldsymbol{x}) = f'\left(\begin{bmatrix}\delta\boldsymbol{r}_{\mathrm{M}}\\ \delta\boldsymbol{v}_{\mathrm{M}}\\ \delta\boldsymbol{\theta}^{\mathrm{M}}_{\mathrm{M}}\\ \boldsymbol{b}_{\boldsymbol{a}_{\mathrm{B}}}\\ \boldsymbol{b}_{\boldsymbol{\omega}_{\mathrm{B}}}\\ \boldsymbol{b}_{\boldsymbol{g}_{\mathrm{M}}}\end{bmatrix}\right) = \begin{bmatrix}\delta\boldsymbol{r}_{\mathrm{M}}\\ \delta\boldsymbol{v}_{\mathrm{M}}\\ \delta\boldsymbol{\theta}^{\mathrm{M}}_{\mathrm{M}}\\ \boldsymbol{b}_{\boldsymbol{a}_{\mathrm{B}}}\\ \boldsymbol{b}_{\boldsymbol{\omega}_{\mathrm{B}}}\\ \boldsymbol{b}_{\boldsymbol{g}_{\mathrm{M}}}\end{bmatrix} + \Delta t \cdot \begin{bmatrix}\delta\dot{\boldsymbol{r}}_{\mathrm{M}}\\ \delta\dot{\boldsymbol{v}}_{\mathrm{M}}\\ \delta\dot{\boldsymbol{\theta}}^{\mathrm{M}}_{\mathrm{M}}\\ \boldsymbol{0}_{3,1}\\ \boldsymbol{0}_{3,1}\\ \boldsymbol{0}_{3,1}\end{bmatrix} \tag{3.57}$$

Where $\boldsymbol{0}_{3,1} \in \mathbb{R}^{3\times 1}$ is a zero matrix of size 3×1. It can be seen that only the errors regarding the navigation solution are updated by the transition function.

3.7.10 Measurement Model

To correct the predicted error state in the UKF update step, the transformed sigma points $\boldsymbol{\chi}_{k+1|k}$ are projected through the observation function $h : \mathbb{R}^{\dim(\delta\boldsymbol{x})+\dim(\boldsymbol{z}_{k+1})} \rightarrow \mathbb{R}^{\dim(\boldsymbol{z}_{k+1})}$. This will compute the gamma points $\boldsymbol{\gamma}_{k+1}$.

Similar to the sigma points, the gamma points have been constructed in a way that each point can be divided into different parts. The first part corresponds to the actual error state vector $\delta\boldsymbol{x}$ and the second part corresponds to the measurement noise $\boldsymbol{v}_{k+1}$. Equations (3.58) to (3.60) show how each part is handled. The state vector part is projected through the actual observation function h', whereas the measurement noise part is added to the predicted observation.

$$\boldsymbol{\gamma}_{k+1} = h\left(\boldsymbol{\chi}_{k+1|k}\right) \tag{3.58}$$

$$= h\left(\begin{bmatrix}\delta\boldsymbol{x}\\ \boldsymbol{v}_{k+1}\end{bmatrix}\right) \tag{3.59}$$

$$= h'\left(\delta\boldsymbol{x}\right) + \boldsymbol{v}_{k+1} \tag{3.60}$$

In the following, the observation functions h' for the different measurement types are given separately. The resulting observation function h' is assembled using a combination of these partial observation functions.

- The modules described in the *Optical Sensor Data Processing* section of this work all output a calculated position estimate $\tilde{\boldsymbol{r}}_{\mathrm{M}}$ with corresponding covariance matrix. To use this output as an input for the sensor fusion, the observation function needs to output a position as well. This is defined in the following:

$$h'_{\boldsymbol{r}_{\mathrm{M}}}\left(\boldsymbol{\chi}_{k+1|k}\right) = \tilde{\boldsymbol{r}}_{\mathrm{M}} + \delta\boldsymbol{r}_{\mathrm{M}} \tag{3.61}$$

- To update the attitude $\tilde{\boldsymbol{q}}^{\mathrm{M}}_{\mathrm{B}}$ of the navigation solution a star tracking camera is used. Based on the acquired images a rotation quaternion $\tilde{\boldsymbol{q}}^{\mathrm{I}}_{\mathrm{B}}$ representing the attitude of the system with respect to the Earth-Centered Inertial (ECI) coordinate frame is computed and fed into the sensor fusion. Assuming the orientation of the MCMF frame to the ECI frame is known at all times, the definition of the observation function $h'_{\boldsymbol{q}^{\mathrm{I}}_{\mathrm{B}}}$ follows:

$$h'_{\boldsymbol{q}^{\mathrm{I}}_{\mathrm{B}}}\left(\boldsymbol{\chi}_{k+1|k}\right) = \boldsymbol{q}^{\mathrm{I}}_{\mathrm{M}} \otimes \delta\boldsymbol{q}^{\mathrm{M}}_{\mathrm{M}} \otimes \tilde{\boldsymbol{q}}^{\mathrm{M}}_{\mathrm{B}} \tag{3.62}$$

- Using the laser altimeter an attitude compensated altitude above the Moon's surface is calculated. This measurement is also used to update the sensor fusion. For the calculation of the corresponding observation function the latitude ϕ of the position $\boldsymbol{r}_{\mathrm{M}} = \tilde{\boldsymbol{r}}_{\mathrm{M}} + \delta\boldsymbol{r}_{\mathrm{M}}$ is calculated first.

$$\phi = \tan^{-1}\left(\frac{\boldsymbol{x}_{\mathrm{M},z} + e'^2 \cdot b \cdot \sin^3(\theta)}{r - e^2 \cdot a \cdot \cos^3(\theta)}\right) \tag{3.63}$$

  Where $r = \sqrt{\boldsymbol{x}_{\mathrm{M},x}^2 + \boldsymbol{x}_{\mathrm{M},y}^2}$ and $\theta = \tan^{-1}\left(\frac{\boldsymbol{x}_{\mathrm{M},z} \cdot a}{r \cdot b}\right)$.
  With that the observation function h'_{alt} is given by:

$$h'_{alt}\left(\boldsymbol{\chi}_{k+1|k}\right) = \frac{r}{\cos(\phi)} - \frac{a}{\sqrt{1 - (e^2 \cdot \sin^2(\phi))}} \tag{3.64}$$

  The parameters semi-major axis a, semi-minor axis b, first eccentricity e and second eccentricity e' depend on the modeled ellipsoid of the celestial body.

Depending on which kind of measurements are available at time k, the dimensions of the measurement vector $\boldsymbol{z}_k$, the measurement noise $\boldsymbol{v}_k$ as well as the observation function h itself are adjusted accordingly.

3.8 Simulation Framework

One of the first steps in the project ATON was to set up a simulation environment which included the dynamics model of a lunar landing vehicle as well as models for all sensors. It is needed in order to create the proper inputs for image processing methods, which are part of the ATON system. For generating artificial images from the given state of the vehicle an extensive simulation was set up. It is described in more detail in the following section. The model uses camera parameters and DEM of the lunar surface. For that purpose the DEM maps of the Japanese Selene mission were acquired and preprocessed [27, 44]. Although the Selene mission provided a global mapping, the DEM resolution is limited. For the final phases of the landing (below 2 km altitude) the noise of the DEM is dominant. For that reason the DEM was enhanced with an artificial structure which can be expected at the landing site [51].

The following section provide more details about the simulation models and the simulation environment.

3.8.1 Simulation of Camera and LIDAR Images

Based on a framework for the simulation of optical sensors [68], a camera simulation was created to provide images and depth information for the ATON processing chain. The goal was to produce physically plausible synthetic sensor data for the ATON software in the loop tests. The optical sensor simulation framework had to be enhanced for ATON to accelerate the creation of synthetic sensor data, so simulations of a large number of camera images could be performed within a reasonable time frame. Additionally, the camera simulation was extended with a closed-loop functionality.

3.8.1.1 Simulation of Optical Sensors

The simulation of optical sensors consists of 3 main steps. The first step is the determination of what is geometrically visible for every sensor pixel. The second step calculates the spectral at-sensor radiance for every sensor pixel, the radiative transfer from a light source through an atmosphere and through the optical system considering scattering at surfaces and the atmosphere. The final step converts the at-sensor radiance into photoelectrons and a digital number for every sensor pixel considering the detector performance and corresponding noises, e.g. photon noise and readout noise. For the ATON camera simulation, the main step is the geometry module as the radiometric calculations can be simplified due to the negligible atmosphere of the Moon and the limitation of the sensor simulation to one spectral channel.

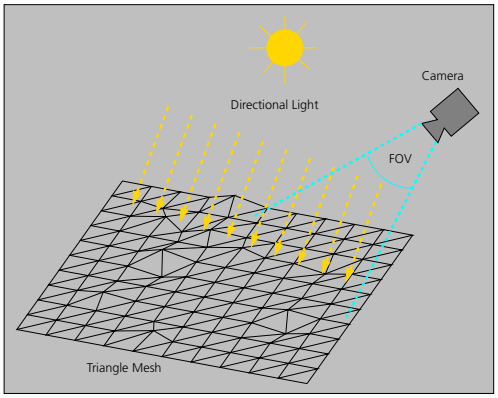


Figure 3.44: Geometrical description of the scene, camera geometry with the field of view, radiometric conditions, and a sensor model are needed for the simulation of optical sensors

3.8.1.2 Geometry Module

The purpose of the geometry module is to determine what every sensor pixel can see, what is shadowed, the surface normal, and the surface material. The hidden surface removal is done by creating $w \times w$ rays,

$$\boldsymbol{r} = \begin{pmatrix} x - \frac{w}{2} + 0.5 \\ y - \frac{w}{2} + 0.5 \\ -\frac{w}{2\tan(\frac{\mathrm{FOV}}{2})} \end{pmatrix}, \tag{3.65}$$

for every sensor pixel, whether one or (in the case of super-sampling) more, and calculating the nearest intersection with the scene geometry, which is described by a triangle mesh, for every ray. Therefore, the scene geometry has to be transformed to the camera coordinate system first. The intersection of the ray $\boldsymbol{s}(t) = \boldsymbol{o} + \boldsymbol{r} \cdot t$ with a triangle ΔABC is calculated in barycentric coordinates,

$$\boldsymbol{T}(u,v) = \boldsymbol{A} \cdot (1 - u - v) + \boldsymbol{B} \cdot u + \boldsymbol{C} \cdot v, \tag{3.66}$$

which have to fulfill the inequalities pictured in Fig. 3.45 to guarantee that there is an intersection point $\boldsymbol{T}(u,v) = \boldsymbol{s}(t)$ within the triangle. The resulting coefficients,

$$\begin{pmatrix} u \\ v \\ t \end{pmatrix} = \frac{1}{(\boldsymbol{r} \times (\boldsymbol{C} - \boldsymbol{A})) \cdot (\boldsymbol{B} - \boldsymbol{A})} \begin{pmatrix} (\boldsymbol{r} \times (\boldsymbol{C} - \boldsymbol{A})) \cdot (\boldsymbol{o} - \boldsymbol{A}) \\ ((\boldsymbol{o} - \boldsymbol{A}) \times (\boldsymbol{B} - \boldsymbol{A})) \cdot \boldsymbol{r} \\ ((\boldsymbol{o} - \boldsymbol{A}) \times (\boldsymbol{B} - \boldsymbol{A})) \cdot (\boldsymbol{C} - \boldsymbol{A}) \end{pmatrix}, \tag{3.67}$$

can also be used to weight the surface normals of the corresponding triangle corners.

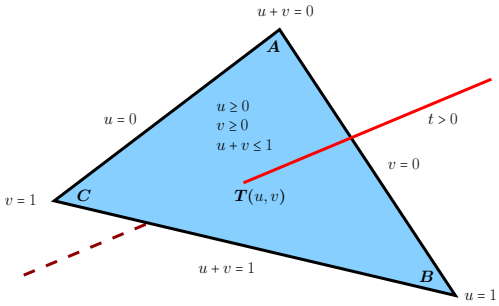


Figure 3.45: Intersection of a ray with a triangle in barycentric coordinates

A depth buffer approach is used to reduce the number of intersection calculations. To calculate whether a pixel is directly illuminated by the light source, the hidden surface removal has to be done additionally from the point of view of the light source. This produces a shadow map which gives binary information about the direct illumination of the geometry part seen by every sensor pixel. As the spatial sampling from the point of view of the camera and the light source differs, an irregular shadow map is needed to avoid aliasing of the shadows seen from the camera.

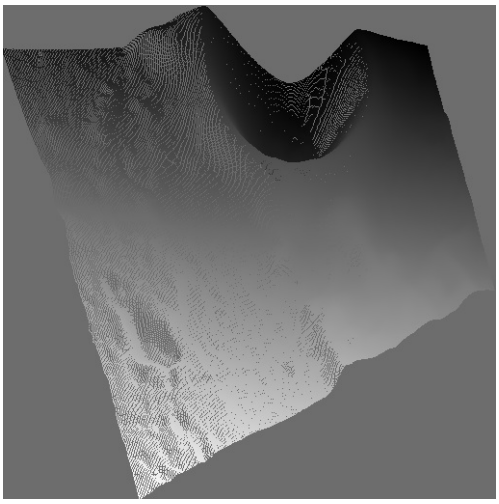


Figure 3.46: An irregular shadow map is used to avoid aliasing during the shadow calculations

3.8.1.3 Radiometry Module

The calculation of the at-sensor radiance can be reduced to the calculation of the direct reflected radiance,

$$L = E_{\text{sun}}(\boldsymbol{\omega}_i \cdot \boldsymbol{n}) f_r(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o) s, \tag{3.68}$$

where

E_{sun} = irradiance of the Sun
$\boldsymbol{\omega}_i$ = unit vector pointing from the surface to the Sun
$\boldsymbol{\omega}_o$ = unit vector pointing from the surface to the pixel of the sensor
$\boldsymbol{n}$ = surface normal
f_r = BRDF of the surface material
$s = 0$ if surface is shadowed
$s = 1$ if surface is not shadowed,

due to the negligible atmosphere of the Moon and the use of the visible spectral range only. The surface material is described by a BRDF. Only physically reasonable BRDFs are used for the sensor simulation. Thus, $f_r(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \geq 0$, the Helmholtz reciprocity $f_r(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = f_r(\boldsymbol{\omega}_o, \boldsymbol{\omega}_i)$, and the conservation of energy have to be fulfilled. To account for the opposition surge effect of the Moon regolith, we use the Lommel-Seeliger law to describe the BRDF that is mixed with a Lambertian part

$$f_r(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = 0.3 \cdot \frac{\rho}{\pi} + 0.7 \cdot \frac{\rho}{2\pi} \cdot \frac{1}{\boldsymbol{\omega}_i \cdot \boldsymbol{n} + \boldsymbol{\omega}_o \cdot \boldsymbol{n}}, \tag{3.69}$$

with the albedo ρ of the surface. Additionally, the at-sensor radiance is distributed to adjacent pixels according to the point spread function of the sensor optics.

3.8.1.4 Sensor Module

The final sensor simulation step converts the calculated at-sensor radiance into the final digital camera image considering the sensor performance (signal to noise ratio). For a quantum detector as used in the ATON camera simulation, the conversion calculates the number of photoelectrons,

$$n_{\text{el}} \approx t \frac{\pi}{4} \frac{p^2}{f_\#^2} \eta L \frac{\lambda}{hc} \Delta\lambda, \tag{3.70}$$

where

t = integration time
p = pixel pitch
$f_\#$ = f-number of the optics
η = overall efficiency
λ = center wavelength
$\Delta\lambda$ = spectral width.

Additionally dark current, shot noise, and readout noise are applied. Finally the quantization step creates a digital number for every sensor pixel. For ATON all sensor parameters are fixed to get well-illuminated camera images.

3.8.1.5 Simulation Input

The main input to the ATON camera simulation is a digital elevation model (DEM) of the Moon. The DEM consists of many tiles which are transformed into a continuous triangle mesh. To reduce aliasing effects due to different spatial sampling of the DEM and the camera, a mipmap procedure is used to re-sample the DEM tiles with different resolutions and the resolution best fitting to the camera perspective is chosen for every tile during the rendering procedure. Furthermore, the position and attitude of the camera and the position of the Sun are needed as input for the simulation. A simple pinhole camera model with 1024×1024 pixels and a field of view of $\mathrm{FOV} = 40\,\mathrm{deg}$ is used. The ATON camera simulation is performed with a 16-times super-sampling to increase the visual quality of the final camera images.

3.8.1.6 Simulation Output

For every camera position and the corresponding attitude and Sun position, the ATON camera simulation saves an 8-bit monochromatic camera image, a 16-bit depth image which can be used to simulate a LIDAR, and the distance from the middle of the camera FOV to the Moon surface to simulate an LA measurement. Moreover, the camera simulation provides a network interface which can be used to create camera and depth images in the loop for desired input parameters.

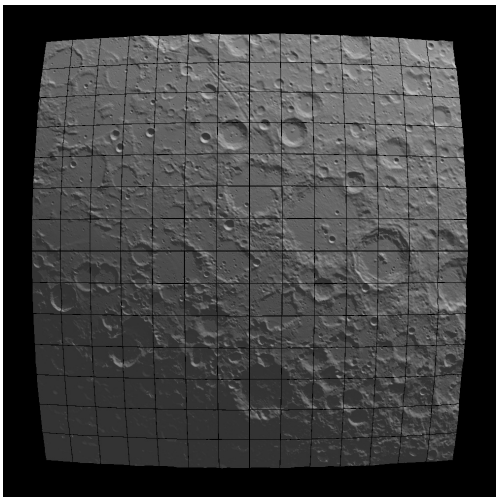


Figure 3.47: The digital elevation model of the Moon north pole consists of 196 tiles. The gaps between neighboring tiles get filled with matched triangles to avoid visible glitches and disturbed shadow calculations

3.8.2 Trajectory and Sensor Modeling

With the goal to develop and verify image processing and navigation methods it is very beneficial if the computation time for simulation models is much smaller compared to the computation time of the techniques to be developed. Therefore in ATON the open-loop simulation used pre-processed data of models which are computationally extensive. Figure 3.49 shows the scheduling for simulation and processing modules in an open-loop simulation. It shows that the trajectory as well as the sensor data of the terrain relative sensors (camera, LIDAR, LA) are pre-computed before the simulation. When running the simulation the time-tagged data are read by the simulation software and are provided as sensor outputs to the processing modules.

All processing modules have been described in more detail in previous sections. The following subsections will provide more information about the simulation models

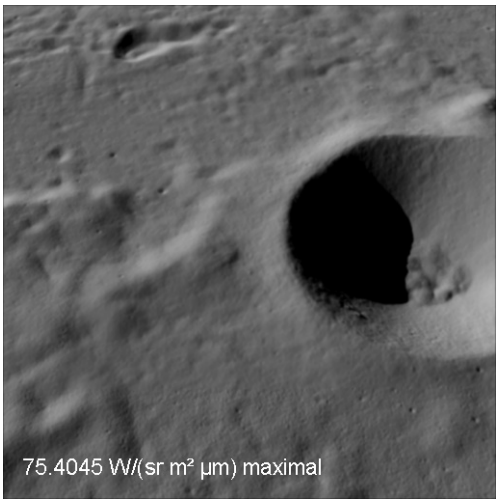


Figure 3.48: A simulated ATON camera image with maximal at-sensor radiance. The irregular shadow map shown in Fig. 3.46 was used for the calculation of the shadows in this image

3.8.2.1 Reference Trajectory for Open-Loop Simulations

In order to provide the necessary inputs for the sensor models a reference trajectory for a lunar landing had to be created. It shall provide position, velocity, attitude, angular rate and specific non-gravitational forces. This information needs to be provided at the highest sample rate of any of the sensors in order to avoid interpolation. In addition, auxiliary information important for the mission, like the Sun vector, is included in the data set.

As already mentioned in Sect. 2.3 the trajectory is a result of solving an optimal control problem. The solution from [65] is used as a reference. For using it in the different scenarios the trajectory is adapted to the chosen landing site and approach direction. It is transformed to the MCMF frame and samples are generated with the required sampling rate.

Since the whole simulation shall start with the DOI a part of the descent orbit must be added before powered descent. In order to match the half ellipse of the descent orbit to the PD trajectory computed with the method described above, position and velocity are backward integrated starting from the PDI.

For the attitude the following definitions and assumptions are made:

1. The attitude at the beginning of the descent orbit is horizontal (body z-axis pointing down, body x-axis pointing horizontal in flight direction).
2. The attitude during the powered descent is defined by a constant pitch angle which can be specified (default 30 deg).
3. The slew time for transition between attitude at DOI and attitude during DO can be specified (default 100 s).
4. The attitude at the end of the DO equals the attitude defined by the powered descent trajectory.
5. The slew time from attitude during DO to attitude at PDI can be specified (default 100 s).

Based on these settings the attitude in the descent orbit as well as the corresponding rates are computed. The two parts of the trajectory are merged and the state vector is formatted to match the input interface of all simulation models.

The state vector is defined as input to the simulation models of the navigation sensors. The vector is split into two parts, the high-rate part and the low-rate part. Furthermore the header contains the following data:

- Date of start of trajectory (in UTC and ephemeris time)
- Sampling rate of high-rate state
- Sampling rate of low-rate state

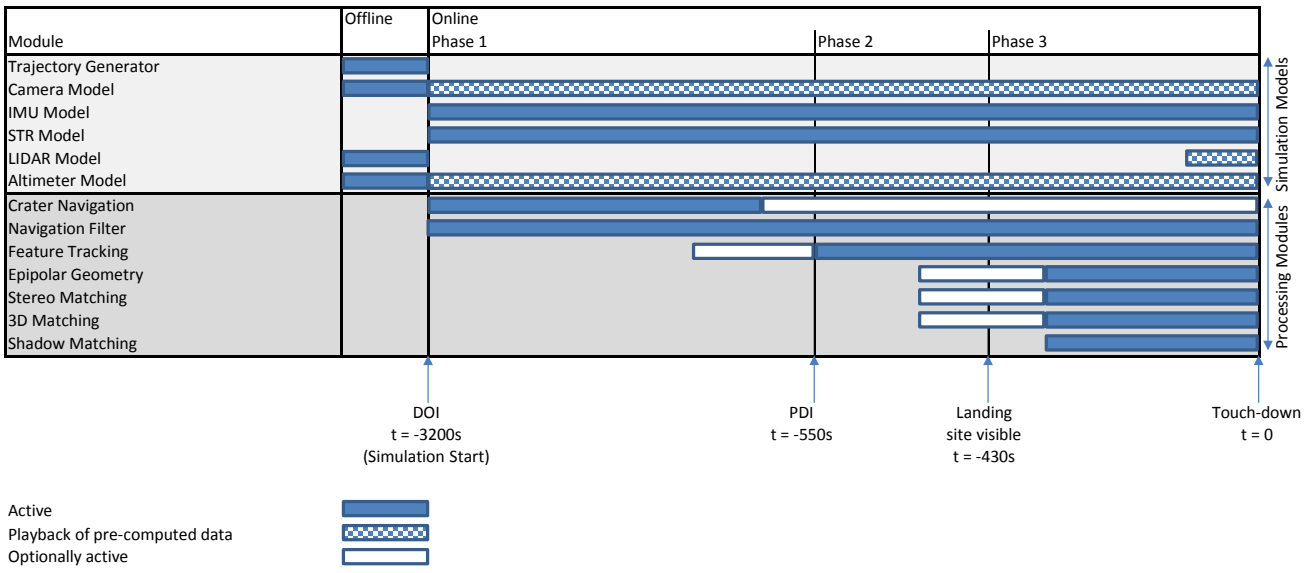


Figure 3.49: Schematic view of the time line of the open-loop simulations in ATON

The low-rate state contains the following quantities in the given order:

No.	Symbol	Description
1	t	Simulation time with $t = 0$ for first state
2	idx	Unsigned integer index (corresponds to milliseconds)
3	r^i	Position in inertial coordinates
4	v^i	Velocity in inertial coordinates (first derivative of r^i)
5	q_i^b	Inertial attitude (denotes rotation from inertial to body-fixed frame)
6	e_{Sun}^m	Unit vector of Sun light in Moon-fixed cartesian coordinates
7	r^m	Position in Moon-fixed Cartesian coordinates
8	v^m	Velocity in Moon-fixed Cartesian coordinates
9	q_m^b	Attitude of vehicle with respect to Moon-fixed frame

The high-rate state contains the following quantities in the given order:

No.	Symbol	Description
1	t	Simulation time with $t = 0$ for first state
2	idx	Unsigned integer index (corresponds to milliseconds)
3	a^b	Inertial acceleration in body-fixed coordinates (second derivative of r^b). It includes the gravitational acceleration (see no. 4).
4	g^b	Gravitational acceleration in body-fixed coordinates
5	$\omega_{i,b}^b$	Angular rate of body-fixed frame with respect to inertial frame expressed in body-fixed frame

Rationales:

- The state vector is used as input for sensor data simulation as well as truth reference for the tested navigation algorithms. Therefore the velocity information has been added although not needed by the sensor models.
- In order to avoid inconsistencies due to mismatch in the coordinate transformation from inertial to Moon-fixed frame position, velocity and attitude is also given with respect to Moon-fixed frame.
- Most sensor data shall be provided at low-rate (max. 30Hz). High-rate is required only for IMU. Thus the inputs required for IMU simulation are put in high-rate state.

3.8.2.2 Sensor Models

This subsection provides a short overview of the sensor models used in the MiL and SiL simulations of ATON.

Camera, LIDAR and Altimeter Models
Since camera, LIDAR and LA are providing measurements depending on the visible terrain their simulation is complex. It is described in Sect. 3.8.1.1. The simulation provides a 1024 × 1024 pixel monochrome image as well as a depth image of the same size. From the depth image a subset of 400 × 400 pixels was cut out to simulate the LIDAR measurements. A single point of the depth image was used for simulating the LA.

In the open-loop simulation the pre-computed monochrome images and LIDAR depth images are loaded by simulation models according to the index provided by the state vector. Similarly the LA model outputs a single distance measurement.

The sampling time for all terrain-relative optical sensors could be changed from the minimum at 1/30s to every multiple of this minimal sampling time.

Inertial Measurement Unit Model
The IMU model is coded in Simulink and represents an IMU consisting of three orthogonal rate gyroscope and three orthogonal accelerometers.

The model reads in the true specific force in the body frame and the true angular rate in the body frame. It then corrupts these values with various error sources, integrates the corrupted values and outputs the back differences as the measured outputs. The IMU model considers the following errors for both the accelerometer and gyroscope:

- Maximum rate
- Bias
- Bias stability
- Scale factor error
- Scale factor stability
- Random walk
- Input axis misalignment
- Limited bandwidth
- Output resolution

Star Tracker Model
This simulation model provides the attitude quaternion of the star tracker body frame with respect to the ECI frame. The output behaves as an output from a typical star tracker. For that purpose it uses internally some models which simulate the attitude measurement errors using the distribution of the stars in the camera focal plane.

The input to the model is the true spacecraft quaternion that describes the attitude of the spacecraft body frame with respect to the ECI frame. The output is a quaternion describing the same transformation but with added errors.

The error model utilizes a star catalog in order to create the expected focal plane star distributions associated with the spacecraft attitude. It considers that the error contribution by a single star increases the further away it is from the focal plane center. In this way lens distortions are accounted for. Furthermore, the number of stars has an impact on the error of the attitude quaternion. The error and noise modeling is based on [81].

3.8.2.3 Adaptation of Models for Closed-Loop Simulation

For proving function and performance in closed-loop operation the simulation was extended by models for vehicle dynamics and actuators as well as a guidance and control function. Furthermore the simulation was connected to the image simulation engine to compute measurements of the terrain-relative sensors (camera, LIDAR, LA) based on the current true state vector which is influenced by the control actions.

The vehicle dynamics were already modeled in the trajectory generations tool as well as for solving the optimal control problem. The model was transferred and encapsulated in a Matlab/Simulink block and incorporated in the MiL simulation.

The guidance and control function have also been implemented in Matlab/Simulink. It is based on a gain scheduled LQR controller which tracks the reference trajectory similar to the design in [65]. The attitude dynamics were not simulated. However, the controller design included a rate limiter which should avoid infeasible rate commands to the attitude control system. Since the trajectory tracking was performed in local downrange, crossrange, and altitude coordinates the needed transformations between these coordinates and the MCMF frame had to be integrated to the MiL simulation.

3.8.3 Simulation Environment

During the course of the project several evolutions of the simulation environment were created to support the different development steps. A Matlab/Simulink environment was selected for the early development. Later, on the target system a new real-time log player was developed. Both environments are introduced in the following.

3.8.3.1 Matlab/Simulink Environment

For the initial development of ATON, Matlab/Simulink was chosen as a platform to integrate the ATON navigation software with the simulated sensors and tools to evaluate the results. The Matlab/Simulink environment was used for the MiL and SiL tests of the project (see Sect. 4.1).

Matlab/Simulink is very well suited to building complex simulations without much effort to connect different modules. Its Graphical User Interface (GUI) allows quick changes to the interconnection of modules and it has a rich set of tools to generate result comparisons, plots, etc., to evaluate (intermediate) results of the system and verify its correctness. The Matlab script language is very powerful for building scripts for mathematical calculations and data conversions.

Fig. 3.50 shows the Matlab/Simulink simulation model that was used for open-loop scenarios. The precalculated state vector (see Sect. 3.8.2.1) is loaded into the simulation and fed into the sensor simulation module (yellow block). The sensor simulation module (see Fig. 3.51) is responsible for loading the prerendered images from the camera and LIDAR simulation and for deriving the sensor outputs from the state vector, for instance, by adding noise to the signals.

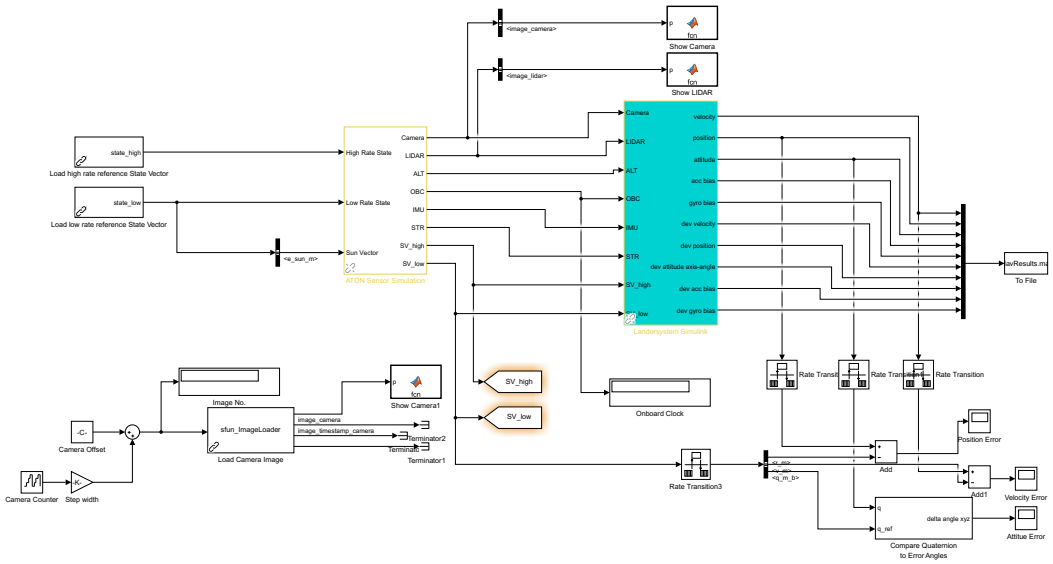


Figure 3.50: Matlab/Simulink simulation model for open-loop scenarios. The sensor simulation of Fig. 3.51 is embedded in the yellow block on the left. The ATON software is located in the cyan block in the middle of the figure. The ATON block can either be the Matlab/Simulink integration (see Sect. 3.9.2 or a s-function with the Tasking Framework integration (see Sect. 3.9.3)

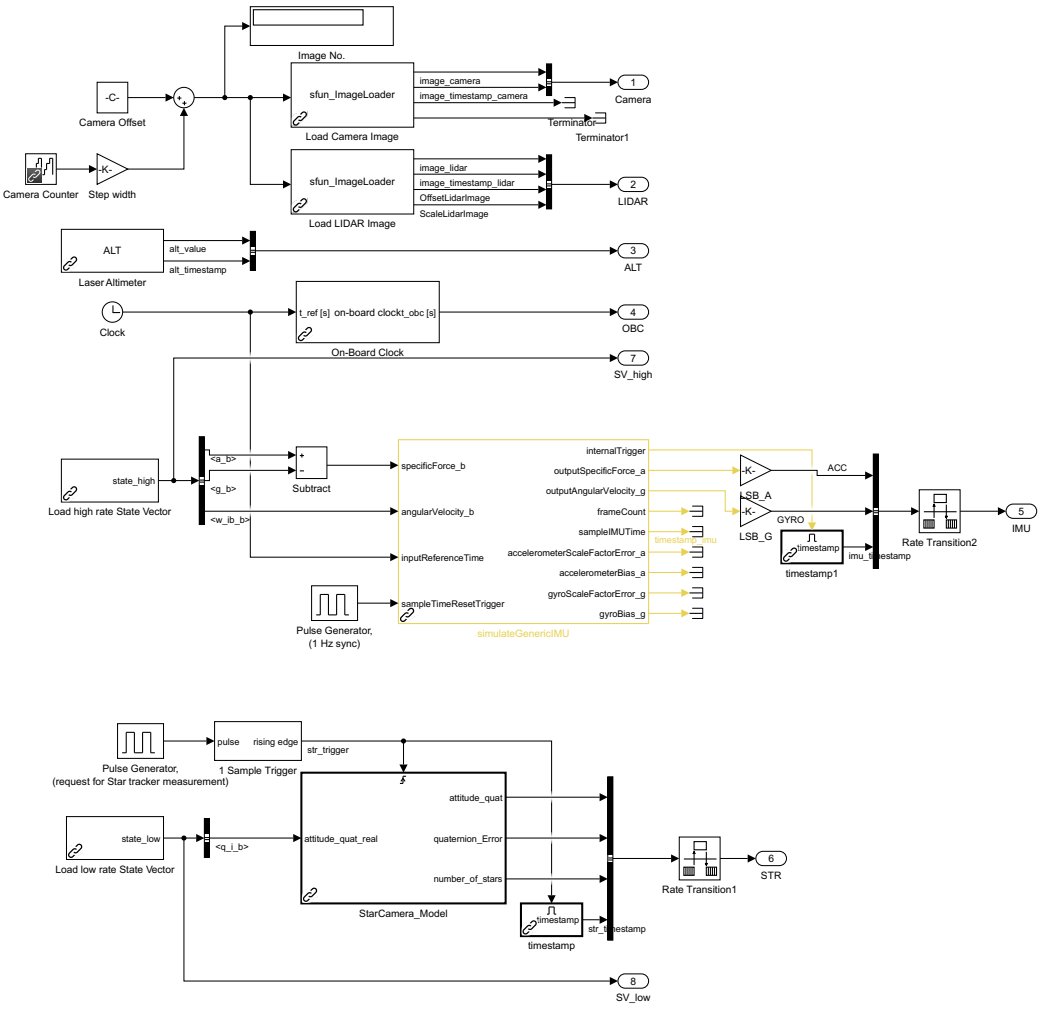


Figure 3.51: Matlab/Simulink sensor simulation model for open-loop scenarios. The camera and LIDAR images as well as the LA values are loaded from disk and fed into the simulation. IMU and STR data are derived from the state vector

The sensor outputs are routed to the ATON model (cyan block), which calculates the estimated position and attitude. The results can then be compared to the truth state from the state vector.

Fig. 3.52 shows the simulation model for closed-loop scenarios. The image shows, besides several blocks for displaying results, a model block (blue) to simulate the landing vehicle dynamics and control. The output of this block is the state vector that is fed into the sensor simulation to close the loop for the closed-loop MiL tests (see Sect. 4.1.3).

The actual trajectory in closed-loop scenarios cannot be precalculated, hence the camera, LIDAR and other sensors have to be simulated during runtime of the simulation. This requires a communication link to the camera simulation application and for the tests in Testbed for Robotic Optical Navigation (TRON) a connection to the laboratory. For this, a network protocol was developed which allows requesting for camera and LIDAR images as well as the laser altimeter measurement for a specific position and attitude of the camera at a given time. Also, the current Sun vector is part of a request to simplify the image rendering in the camera simulation.

The protocol is a client/server protocol implemented with the ZeroMQ[4] network library. It uses the common TCP/IP protocols on the transport and network layer, which allows the client and server to be run on different computers. On the application layer, a custom Extensible Markup Language (XML) protocol was defined.

Fig. 3.53 depicts the schematic of the protocol. The simulation stops and requests the data from the server by a *RequestMessage*. The server then generates both images and the laser altimeter measurement – in case of TRON only a camera image is created – and returns an *ImageMessage* or

[4] http://zeromq.org/

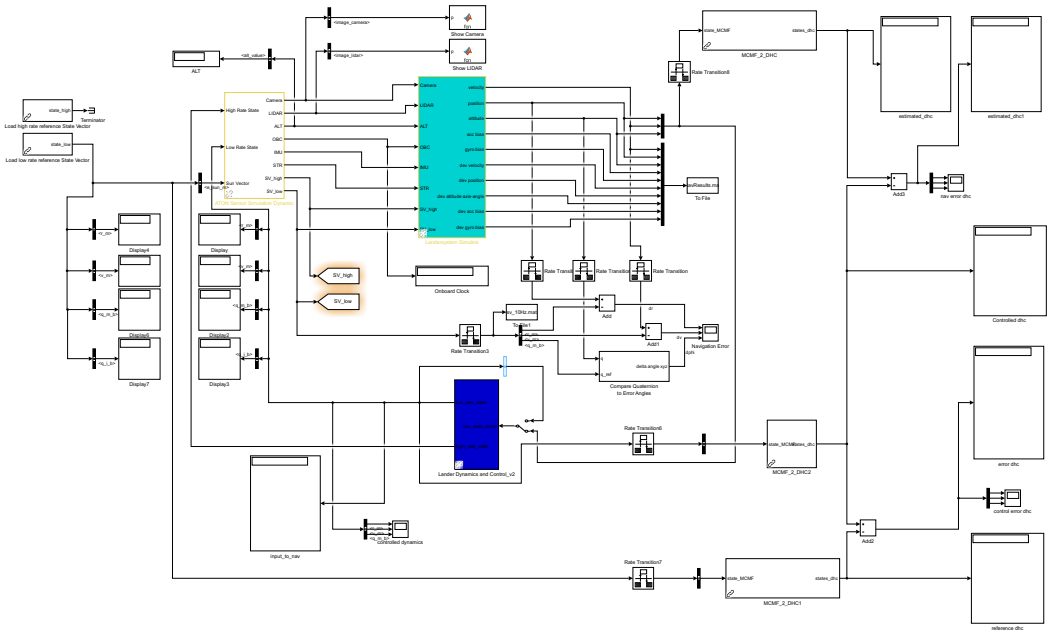


Figure 3.52: Matlab/Simulink simulation model for closed-loop scenarios. The sensor simulation of Fig. 3.54 is embedded in the yellow block on the left. The lander dynamic and control simulation is located in the blue block. The ATON software is located in the cyan block in the middle of the figure. The ATON block can either be the Matlab/Simulink integration of (Sect. 3.9.2) or an s-function with the integrated Tasking Framework (see Sect. 3.9.3)

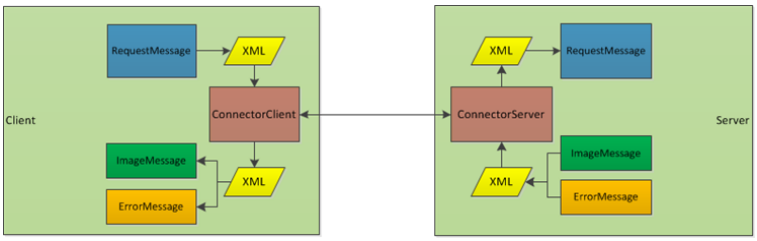


Figure 3.53: Schematic of network communication between the Matlab/Simulink simulation (Client) and image sources like the camera simulation (see Sect. 3.8.1) or TRON (Server)

an *ErrorMessage* if something went wrong. In the case of a successful image delivery the simulation continues.

In the simulation environment the network client is realized as an s-function which replaces the image loader of the open-loop sensor simulation block. Fig. 3.54 shows the sensor simulation block for the closed-loop scenarios.

With all the benefits of this approach, Matlab/Simulink is not intended to simulate in real-time. When a block is executed the simulated time is stopped until the module provides its result. The Matlab/Simulink ATON environment was not used to evaluate ATON regarding real-time performance.

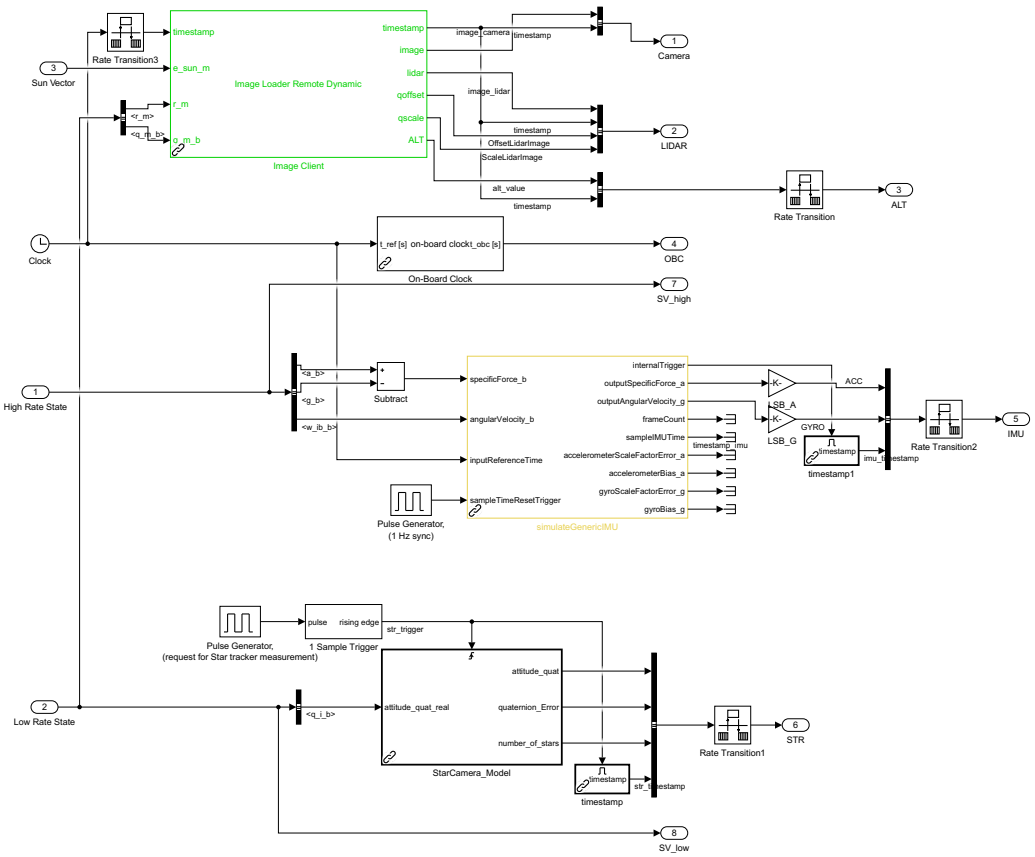


Figure 3.54: Matlab/Simulink sensor simulation model for closed-loop scenarios. The camera and LIDAR images as well as the LA measurements are remotely acquired from the camera simulation (see Sect. 3.8.1) or TRON (see Sect. 4.2.1) by the network client (green block) and fed into the simulation. IMU (yellow block) and STR data are derived from the state vector as a result from the closed-loop simulation

3.8.3.2 Real-time Log Player

For Processor-in-the-Loop (PiL) tests, an environment was needed to simulate the input of the sensors to the ATON system in real-time to evaluate the real-time behavior of the software. We developed a logging/log player framework for this purpose. All sensor data, whether images or the other sensors, have time tags. The log player replays a trajectory by sending the sensor data when its time tag is reached. This way, the ATON software is stimulated the same way as it would be by real sensors. The architecture is similar to Fig. 2.11 on Page 23 with the exception that the sensor blocks are replaced by the real-time log player components. Hence, it is a distributed log player where each sensor is represented by its own playback software module.

As a first step, we converted the open-loop scenarios from Matlab/Simulink to our log player format and tested that way the performance of the software under real-time conditions in open-loop. The next step was to develop a logging framework to gather data from the sensors that were used for the flight tests. The sensor data was logged with two time stamps. The first time stamp represents the time when data is recorded at the sensor, for instance when the camera is triggered. The second time stamp notes the time when the data is first processed in the driver on the computer, hence, it is visible for the first time in the ATON software. This second time stamp is used in the log player to feed data in the system. This allows us to emulate delays that are observable on the real hardware.

A big advantage of the logger/log player approach is the possibility to directly replay the collected data from a flight or any other test without a data processing step. The disadvantage of this approach compared to the Matlab/Simulink environment is that a simulated scenario in the log player cannot run faster than real-time because it is bound to the real-time clock of the target computer. Matlab/Simulink, on the other hand, executes a simulation as fast as possible.

3.9 Software Integration

The ATON software combines the previously introduced processing modules in a software application with sensor data as inputs and the estimated navigation state as output. Since ATON relies extensively on image processing and high frequency sensor data processing, the execution platform for ATON has to fulfill ambitious requirements concerning real-time constraints, parallelization and handling large amounts of data.

The Navigation Filter needs to be executed with 100 Hz to provide constant updates to the control systems of the vehicle. On the other hand, the different image processing modules have varying runtimes to process an image. The runtime ranges from a few milliseconds to several seconds. This different and non-deterministic runtime behavior of the modules is a challenge that needs to be addressed when building a real-time capable system.

Moreover, different institutes and development teams have used varying coding styles, guidelines and even different programming languages for their modules because not all algorithms were developed solely for this project [19]. This made the integration quite challenging throughout the project.

In the following, the software architecture is presented. Then, the iterative integration into the Matlab/Simulink environment (see Sect. 3.8.3.1) is described in Sect. 3.9.2, followed by the integration on the target platform based on the Tasking Framework (Sect. 3.9.3). In Sect. 3.9.4 the Model-Driven Software Development (MDSD) approach is presented, which was introduced midterm during the course of the project.

3.9.1 ATON Software Architecture

The high-level view of the ATON software, as can be seen in Fig. 2.11, is that of a filter. Several sensor inputs are fed into the software, the input data is processed by the modules, finally the results are fused by the Navigation Filter and the result is output as navigational result. This means ATON has a data flow based architecture. Further analyses and development showed that on the top level virtually no control flow has to be considered. Except, in some milestones, certain modules were switched on or off for different phases of the landing.

Furthermore, during the course of the project it became apparent that external commanding of ATON was also very limited. Only some commands are necessary to control several initial steps of the flight experiments. For other scenarios (SiL, HiL, PiL), no commanding at all is required.

To build the data flow architecture for ATON and to allow modularization of the different processing algorithms, a high-level interface for all modules was defined. All modules have to provide inputs, outputs, and parameters. This is the only interface and interaction of a module to its environment. New input data triggers the execution of one module and it produces one or several outputs. The parameters are defined before the software is started and not changed during the execution of the software.

The scheduling of the modules is also driven by the data flow. A module should only be executed when all necessary input data is available.

The data flow within the system has to be distinguished in two classes: reliable data flow and data flow with the discarding of old data. Several sensor readings should be consumed completely by the modules since losing updates would decrease the performance of ATON. Other data flows should allow for discarding of older data since some modules have a processing time for an update that is longer than the frequency of their input data. In this case the software should provide only the newest input data and should discard older data.

We have also to distinguish two types of data: high volume image data and other low volume sensor data like star tracker or IMU measurements. The images taken from the cameras or LIDAR are comparably large: 1024×1024 pixels with 8 bit for the cameras and 400×400 pixels with 16 bit for the LIDAR depth images.

As an example, Fig. 3.55 shows the ATON software in the configuration for the closed-loop flight test (see Sect. 4.3) as a System Modeling Language (SysML) internal block diagram. The blocks represent a module of the system that communicates with other modules via the depicted links. On the right side the data logger is shown. In between are software modules and some helper modules which act, for instance, as sample rate converters or as communication Interface (I/F) with the helicopter (a detailed description is presented in Sect. 4.3.2).

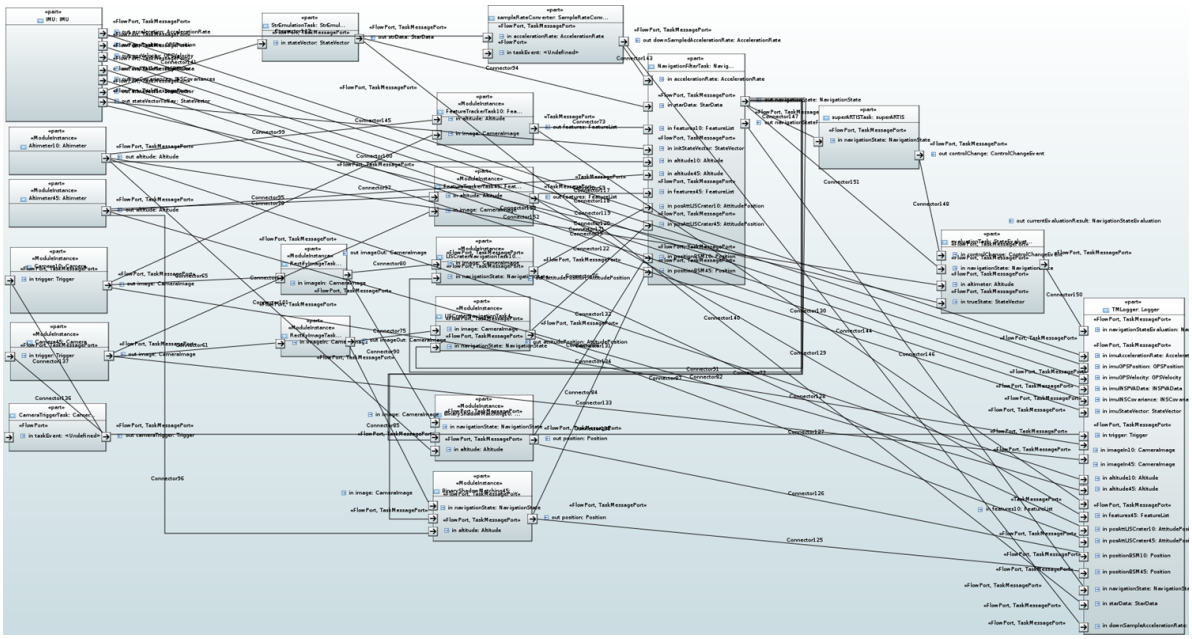


Figure 3.55: SysML internal block diagram of the ATON software for the flight tests

3.9.2 Matlab/Simulink Integration

The initial approach to combining the different algorithms was to use Matlab/Simulink [19]. We modeled the navigation system with a block diagram and integrated software modules via s-functions. The algorithms are implemented in C or in C++. This approach was used to develop and adapt the different software modules in a sensor simulation environment. The behavior of the modules was not modeled, as already mentioned, since the algorithms were partially developed in other contexts. Fig. 3.56 shows the integration model with the different modules. The 3D chain is integrated in a submodel as shown in Fig. 3.57.

The integration of independent modules in Matlab/Simulink allowed a more or less independent development of the different modules. It is also easier to test single modules independently. The environment also allowed for simple access to module results.

3.9.3 Tasking Framework Integration

In the following, we introduce DLR's Tasking Framework followed by a description of the integration of the ATON software modules with the Tasking Framework.

3.9.3.1 Introduction

To realize the requirements and the above described software architecture on a real-time system, DLR's Tasking Framework [57] was used and extended to integrate all of ATON's software modules. In traditional embedded control or navigation systems all necessary computations are started on an expectation of the sensor timing. The expectation of the sensor timing is conservative, which reduces the timing window for the computations to keep the real-time requirements of the system.

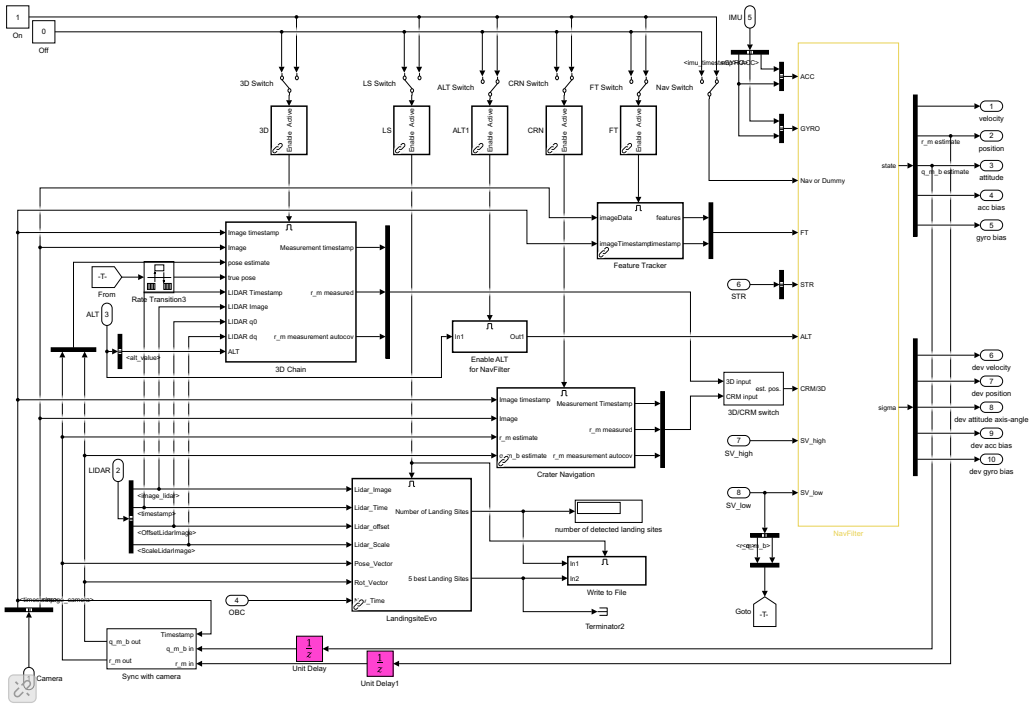


Figure 3.56: Integration of the ATON modules in the Matlab/Simulink environment. Each module is realized by an s-function. The 3D chain is depicted in Fig. 3.57

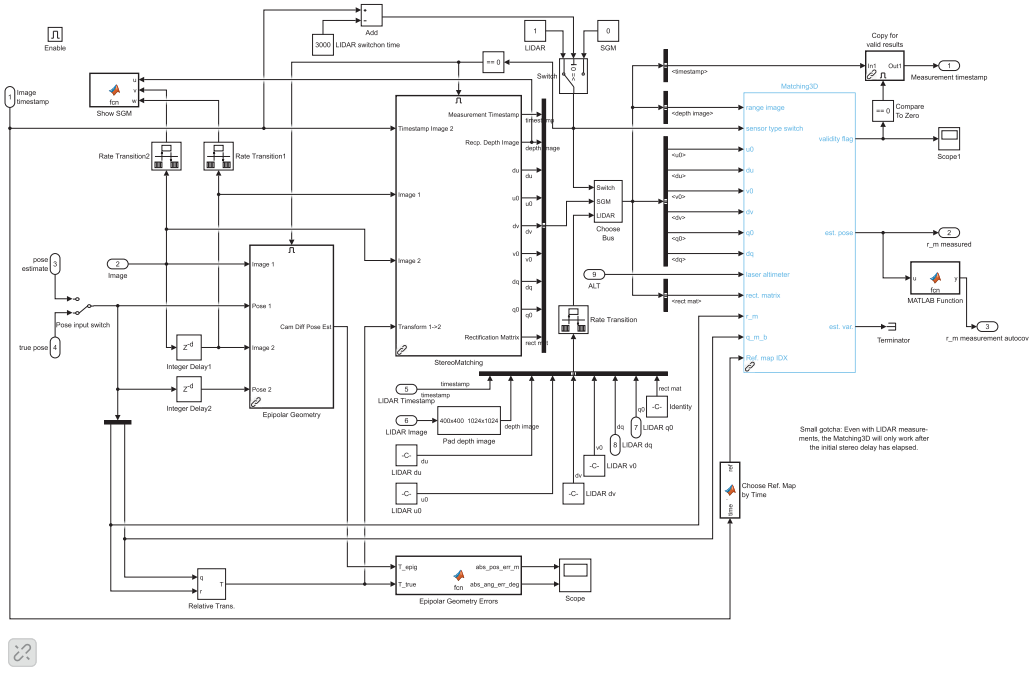


Figure 3.57: Matlab/Simulink integration of the 3D chain

The Tasking Framework provides the means to drive the computation tasks in a reactive way instead of by using a fixed timing. Reactive means that a computation task starts when all available messages at its inputs are available or an event triggers the computation task. Messages in this context are data represented in memory. These data representations are the result of reading messages from a device or the network by a device driver or any other computation task.

The messages are organized in so-called channels. A channel is realized as a single data structure, a queue, a ring buffer, etc. Tasks can create or consume data directly in or from a channel. This reduces the need for copy operations. This means a message, which is sent from one task to another, is not actually transmitted or copied; only the reference to the data is passed to the receiving task.

An event is not associated to any data. For instance, an event could be a clock tick. Messages and events can have a desired quantity to trigger the computation, for example the number of elements in a queue.

The Tasking Framework supports grouping of tasks in task sets. In such a task set each computation task is only executed once until all other tasks have been executed. Priorities can be assigned to tasks as well.

To satisfy real-time requirements it is also possible to execute a task if not all messages and events are available by marking a message or an event as final. This is a usually a time-out event that ensures task execution before a certain deadline.

The Tasking Framework is implemented in C++ and currently supports Linux and Real-Time Executive for Multiprocessor Systems (RTEMS) as its operating system. The RTEMS port is based on DLR's OUTPOST platform[5]. The Tasking Framework provides executor threads to which its own scheduler assigns tasks depending on their state and their priorities. Usually, one executor thread per available processor core is used.

With the concepts of tasks, events, channels and messages the Tasking Framework is well suited to implementing ATON's data flow oriented, event-driven software architecture. Fig. 3.58 shows a simplified example of a navigation system. It consists of two sensors (*Star Tracker* and *Camera*), an image processing task (*CraterNavigation Task*) and a *NavigationFilter Task*. The *Attitude* messages from the *Star Tracker* trigger the *NavigationFilter Task* since its input is set to 1. The *Position* message from the *CraterNavigation Task* a considered as optional (input set to 0). This means they are considered by the *NavigationFilter Task* if available but the task is executed even if no new *Position* message is present. The *CraterNavigation Task* in this example is executed when at least one *Estimated Position* and 30 *Image* messages are available. This does not mean the task has to process 30 images but that it is only started for every 30th image.

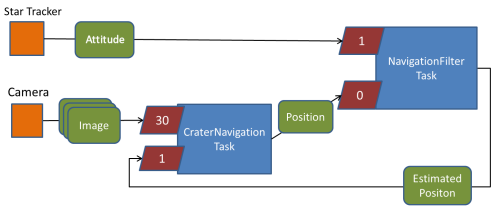


Figure 3.58: Simplified example of a navigation system based on the Tasking Framework. The *NavigationFilter Task* is executed every time a new *Star Tracker* message is received. The *CraterNavigation Task* is only executed if at least one *Estimated Position* message and at least 30 *Camera Image* messages have been received

3.9.3.2 Integration

To integrate the different software modules into the Tasking Framework we defined an interface for the software modules that is similar to the programming interface of Matlab/Simulink's s-functions. This allowed the sharing of code of the modules between integration platforms. Only the

[5] OUTPOST - Open modUlar sofTware PlatfOrm for SpacecrafT - https://github.com/DLR-RY/outpost-core

interface code for the different platforms (s-function and Tasking Framework) had to be adapted.

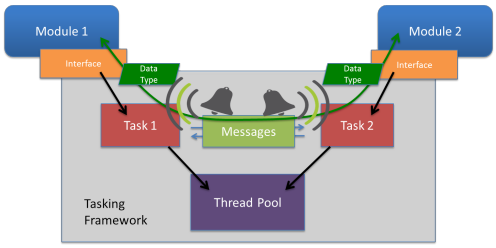


Figure 3.59: Illustration of the module integration into the Tasking Framework. The modules have a standardized interface that uses globally defined data types. Each module is executed in the context of a task that is responsible for exchanging messages with other tasks. Tasks are organized in task sets and executed by a thread pool

Fig. 3.59 depicts the general architecture of the integration into the Tasking Framework. A software module is embedded into an interface that is executed in the context of a task. The data types for the messages are globally defined and any necessary data type transformation is realized in the interface. The task is responsible for handling the communication with other tasks. When a task is executed, it prepares the received data for the module, calls the module and sends the result to other tasks.

Two different channel types are used in ATON: First-In, First-Out (FIFO) and so-called synchronized messages. The FIFO channel provides the classical queue semantics with the addition of supporting multiple readers. This means several readers can read the contents of the FIFO and the channel logic ensures that the FIFO semantics are preserved for all receivers. The multiple reader implementation does not require that the message data has to be copied for each reader.

The synchronized message channel is used when not all data on a channel can be consumed. For example, the Crater Navigation task cannot process every image of the camera. If a FIFO were used, the Crater Navigation task would need to consume every image message in the FIFO until the latest is reached. All prior images have to be dropped because only the latest image should be processed. The synchronized message channel always delivers the most recent message and drops older messages if they are not currently being processed by any task. This synchronization ensures consistency of data when it is currently processed.

3.9.3.3 Logger

In ATON a special logger is used to handle the high volume of sensor and processing module data. It makes use of task messages to be as generic as possible. The logger works in two steps: filtering and writing.

An incoming message is first fed through a user-defined set of filters. The filtered result is then fed to a user-defined set of writers. The filters can be used, for example, to reduce the frequency of a message type, select only a subset of data entries in a message or reduce the size of an image. Depending on the filtered result, specific writers process the result further. There are writers to save images to the disk, to create log files and to send telemetry via a network collection. The actual serialization of all data, except binary image data, is part of the global data type definition. This way, the logger is implemented in a generic way and new message sources and message types can easily be added to it.

The logger architecture allows for the handling of the same data differently for different targets in parallel, whether it is high-volume local logs or telemetry with a limited bandwidth.

3.9.4 Model-Driven Software Development (MDSD)[6]

The first versions of the ATON software were manually integrated with the Tasking Framework. The integration of all these modules into a working software turned out to be very challenging. Besides defining software interfaces and data types in an Interface Control Document (ICD), several misunderstandings occurred during integration. Since the corresponding module developers provided the interfaces, data types had different formats and needed to be converted during integration. Therefore, the intermediate step to integrate the modules first in Matlab/Simulink turned out to be the right approach. In that environment it was easier to test the modules independently.

Additionally, the integration of the software modules into the Tasking Framework as the execution platform was very time-consuming. The interface and tasking communication logic for each task needed to be manually and individually coded. Besides the actual integration of a module, changes in an interface caused much work as well. Every modification needed to be documented in the ICD and we had to update all related parts of the software.

To overcome the before mentioned integration problems, we introduced a Model-Driven Software Development (MDSD) approach. The general idea of MDSD is to collect all required design data for information exchange between engineers in a central model rather than using documents. Instead of trying to combine interfaces implemented independently, the coherence of the components and the software's internal interfaces should be defined from a system point-of-view. Defining this kind of information in a formal model enables analysis and reduces misunderstandings between all involved parties [89].

Models can be used to support design, analysis and validation activities even before the software implementation. For instance, with an ICD the compatibility of inputs and outputs of communicating software modules can only be checked manually. If a formal model is used to define inputs and outputs, a software validator can automatically check for compatibility. Since modeling environments are usually based on standardized concepts like the Meta-Object Facility (MOF), models can be analyzed, validated and transformed with existing tools [10].

Model-driven development refers to the process of formally describing a system in a model and then generating project artifacts from it [11]. As Fig. 3.60 shows, Model-Driven Software Development (MDSD) is part of the model-driven development and uses the model to generate source code, documentation and unit-tests. The main motivation of using model-driven development is to increase productivity [7]. Short-term productivity is improved by generating new features from the model. Long-term productivity rises because changes of requirements can be easily handled by changing the model. To avoid models with redundant content, systems are described independently from target platform and desired programming language [11].

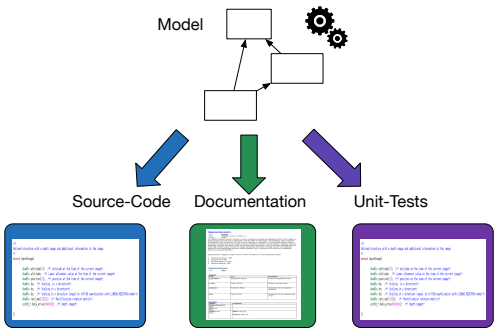


Figure 3.60: Model-Driven Software Development (MDSD) in ATON: generating source code, documentation and tests from a single data model [19]

In ATON we decided to apply MDSD to the structural parts of the software to overcome integration problems and to reduce the overhead for adding

[6] This subsection is based on [19].

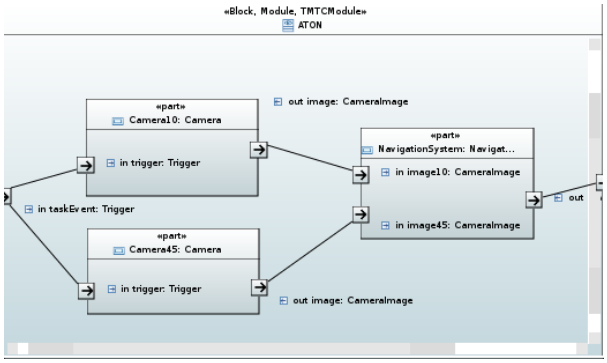


Figure 3.62: A simple example of an internal block diagram: an external trigger synchronizes two cameras. The navigation system computes a navigation solution and provides it to an external interface [19]

and removing new software elements. Because most algorithms were already available as software modules when MDSD was introduced, we did not describe their behavior within the model.

3.9.4.1 Modeling

Generating C++ classes from the system model addresses the goal of achieving consistent interfaces. In addition, it can reduce the overhead for adding new modules into the system, since the boilerplate code is automatically generated. Moreover, the source code to establish communication channels and execution containers reduces the development overhead significantly. Especially if new software modules are added or removed, no manual coding is necessary to adapt the communication logic. To generate this kind of code, the model needs to represent the communication and its parameters. Usually, diagrams are used to represent this information for complex systems. Blocks and lines are easy to understand and directly depict the data flow.

The model is a formal abstraction of the system under development. It is the basis for building the system architecture and the generation of source code as well as documentation. We decided to use and extend well-known modeling languages for our purposes. We selected a combination of UML and SysML. Fig. 3.61 shows the relation between both languages.

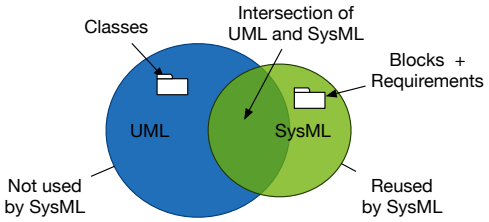


Figure 3.61: Illustration of the relation between Unified Modeling Language (UML) and SysML. The concept of classes is contained in UML, while blocks are part of SysML. Figure from [19] and inspired by [37]

The underlying model orders the contents hierarchically and can contain multiple diagrams. Diagrams provide different views of the system. UML and SysML editors do the actual graphical representation. We used Papyrus, an Eclipse-based editor [23]. It creates the models by using a native UML implementation, which is based on the Eclipse Modeling Framework.

The main diagram provides an overview of the system and describes the data flow. It is implemented using an internal block diagram offered by SysML. The root element of such a diagram is a block representing the system, in this case the ATON software. Subcomponents are added to the model as a block while their diagram representations are parts. Separating between a block and a part brings an important instance-of relation. One software module can be added twice to a diagram, consuming different input values.

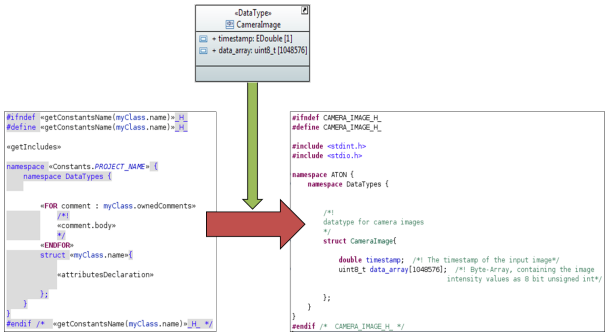


Figure 3.63: Example of the code generation process: source code templates (left) are filled with information from the model (top) to generate the final C++ source code (right)

Fig. 3.62 shows a simple example system with two cameras, sending their images to a navigation system. It computes a navigation solution and provides it as an external interface to the system. The small squares appended to the parts are ports. They represent a communication endpoint between two elements. Ports can have a type and other communication related parameters, which are necessary to generate their source code.

While the internal block diagram provides an overview of the software and its communication, the actual data structures are specified in another diagram. Data types are simple classes with attributes, so UML class diagrams are an appropriate way to define them. Besides the data structures, the parameters of the software modules are modeled using class diagrams.

With the definition of incoming and outgoing data, their structures and the component parameters, all information for the internal interfaces is modeled and no additional explicit definition of interfaces is necessary.

Besides the general structure specified by UML and SysML diagrams, special notification and execution parameters need to be added to the model. To create a profile for language extensions, a UML profile diagram must be created. The element to be customized has to be imported as a metaclass and can then be extended by a stereotype. Stereotypes are classifiers which can contain tagged values, constraints and custom icons. Parameters like the component priority and notification configuration are added as tagged values.

3.9.4.2 Code Generation

After the creation of the system model, it can be used to derive project artifacts from it. The generation is done with Xtend, a language compiled to Java and providing a powerful template feature [15]. The syntax is intuitive and it is possible to debug the templates comprehensively. The creation of custom templates enables the generation of source code for any language. Fig. 3.63 depicts the general process of code generation.

The source code generator is implemented using a combination of the decorator pattern [20] and the generation gap pattern [18]. The motivation to unify interfaces and to reduce the overhead for new modules leads inevitably to the generation of interfaces. However, interfaces are always the boundary of two subsystems, in this case, the generated communication code on one side and the manually implemented code of the software modules on the other side. Mixing the module's functional source code directly into the generated code has its disadvantages because this would

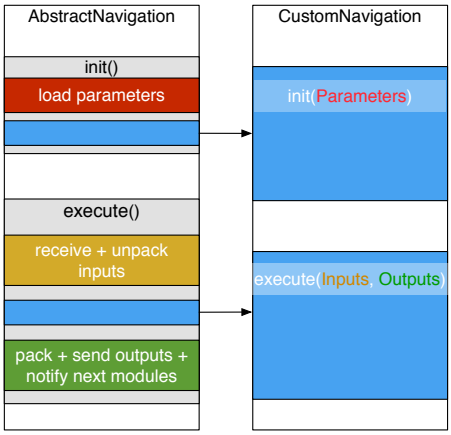


Figure 3.64: Combination of the decorator and the generation gap pattern: the generated class (left) calls methods in the customized class (right) [19]

mean mixing generated and manually written source code. If the model is changed, the source file is regenerated and the functional code has to be manually added again. This would reduce the benefits from MDSD dramatically.

The generation gap pattern solves this problem by providing a solution to combining manually implemented and generated code. The idea is to generate the source code in one class and perform customizations in another. The class for the manual adjustments inherits from the generated one to benefit from the generation. The automatically created classes are stored in a special folder, which should not be edited manually. However, an interface class for the module developer is generated during the first run and later ignored by the generator. This approach enables the addition of module code to the interface class without the need to create a new file.

To reduce the overhead for adding new modules, the generator also creates the communication code for the execution platform. If a module is executed, its parameters and inputs have to be loaded. Fig. 3.64 shows how the generation gap pattern is combined with the decorator pattern to load and send data between system modules. If a module is triggered, the scheduler calls the execute method of the execution container. Its generated code receives the input data, unpacks it and provides it to the class containing the custom module code. After the delegated method in the custom class has finished its execution, the generated method packs the outgoing data and sends it to the succeeding modules.

This implementation differs from the usual decorator pattern. Usually, when using a decorator for extending a class, the extension class inherits from the base class. In this case, the base class is generated and the extensions are located in the manually written subclass. The delegation uses pure virtual methods to call the subclass.

Besides communication code, the configuration management is also generated. If a module is going to be initialized, the generated code calls a configuration manager, which parses a configuration file. The model contains the parameters. This has the benefit that a module developer does not need to take care of parameter definition or loading during initialization of his or her software module. The MDSD framework handles this by generating the source code for parsing a central configuration file.

The MDSD framework also allows modules to be attributed as sensors. These sensor modules support two implementation variants. For flight

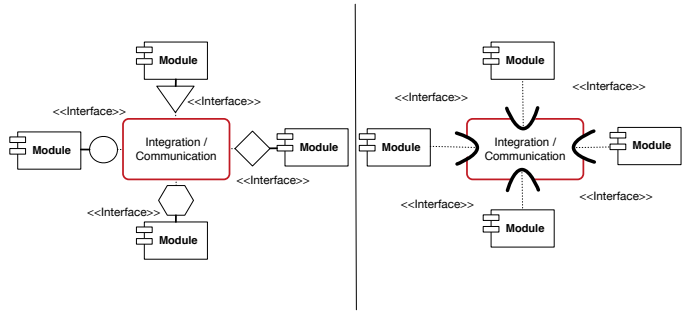


Figure 3.65: Interface communication before (left) and after (right) the introduction of a system model to the software development. The central definition in a model leads to a shared data type over all interfaces. Code generation unifies structure and coding style [19]

tests they consist of the sensor driver to receive the sensor data and preprocess it. The second set of implementations, generated by the framework, consists of the log player components. With this approach, a simple compile flag results in either a flight system software or a log player software configuration. This is achieved by also automatically generating serialization and deserialization logic from the modeled data types. The framework supports a set of basic data types (e.g. integers, floats, strings, arrays, enumerations, etc.) and enables combinations of these to complex data types. The serialization and deserialization logic is also used for sending and receiving telemetry.

Because changes in the system design do not only affect the source code but may also change the documentation and infrastructure, it also makes sense to generate as many of these artifacts as possible.

To create documentation for the generated interfaces, we generate Latex files, which can then be used to create a PDF file. The generated documentation contains the description of all model elements as well as tables for in- and outgoing data, their types and the parameters of the module.

Because the generation gap pattern introduces a set of new classes, the build configuration gets more complex. To solve this problem, the generator also creates a build system file for each module, which provides variables with the necessary include and source files.

Besides documentation and build-system files, the generator also creates unit tests. The integration of tests into the generator brings the benefit that errors caused by unsupported elements from UML or SysML are identified. For example, the unit test for the configuration loader would fail if a parameter type is either not supported by the code generator or by the underlying library.

With the definition of communication channels in the system model, it is possible to generate automatic tests that validate exchange of data between the module interfaces. Generated implementations of the modules send and validate received data.

3.9.4.3 Results

The application of model-based approaches had a positive impact on the ATON project. With the generation of the interfaces and its data structures from templates the intended goal of unification was reached. One template for all interfaces results in classes with the same structure, coding

style and naming conventions. Figure 3.65 shows the difference before and after the introduction of MDSD to the ATON software framework. During the manual integration, module developers defined the module interfaces and the integration and communication code needed to contain the conversion of data types. With MDSD, the generator creates the module's interface code using shared data types defined in a central model. Furthermore, the central definition of data structures for all modules reduced misunderstandings in the development team and also reduced the need to convert the data within the interfaces.

Furthermore, the effort for the software development is reduced significantly. While changing the interfaces or data types has previously been associated with various changes in the source code and documentation, it is now a simple task. After the change has been applied to the model, the generator can then update all related files. In particular, the complex communication and execution code need to be implemented only once, and then the template can be applied to all software modules and communication channels. This way, the development benefits twice: it has become less work to integrate new modules and in the end, changes either in the model or in one of the used libraries can be solved by doing the adjustments in only one place. For example, if the method for sending messages between modules changes, it is sufficient to alter the execution container's template. No manually written code needs to be updated.

Additionally, it is possible to create several configurations for different requirements. The system for a flight test on Earth with an unmanned aerial vehicle needs different modules than a scenario simulating a landing on the Moon. To solve this challenge, it is possible to create different models, one with hardware drivers for flight tests and one with the sensor simulation for a lunar mission. In this way, it is possible to use the same modules for different scenarios by only generating code from different models and using different configuration files.

In the course of the project, model changes became a regular development task. In the integration phases, small changes like updates of data types or parameters appeared on a weekly basis. The addition or removal of communication channels was rarer but its consequences on the source code were larger. Without the code generators, the developers would have had to have cared about memory management, the event-driven execution of the tasks and data exchange between the software module's threads. Furthermore, design changes of the navigation system usually required adding or removing software modules.

Nevertheless, the development team did not only benefit from code and document generation, but the communication between engineers working in different domains also improved. Before, interfaces and its documentation could be changed without realizing the potential impact to the rest of the system. With the model, on the other hand, changes are directly evaluated which, for example, immediately reveals incompatible types of communication channels. Thus, if types need to be changed, the affected interfaces are updated and the developers become immediately aware of the changes.

4 Milestones

Several milestones have been achieved since the start of project ATON in 2010. These can be grouped into four phases:

1. Setup of simulation environment including the simulation of images of the navigation camera and LIDAR as described in Sect. 3.8.
2. Integration and verification of software modules in a MiL environment and later a SiL environment.
3. Verification of the navigation system and elements of the system in HiL and PiL test environments.
4. Verification of the navigation system with outdoor flight tests using an unmanned helicopter testbed.

4.1 Model-in-the-Loop and Software-in-the-Loop Tests

4.1.1 Model-in-the-Loop Tests

For initial development – and also for verification in later development stages – the image processing and navigation modules have been embedded in a Matlab/Simulink-based simulation environment (see Sect. 3.9.2). Since most of the modules have been based on C/C++ coded processing libraries the same coding language has been used. For testing the modules their code was embedded in Matlab/Simulink s-functions. The sensor models for STR and IMU have also been created in Matlab/Simulink. As described above the simulation of images is a very extensive task. For that reason the camera and LIDAR models in Matlab/Simulink were just loading precomputed image files into the simulation. Thus the initial simulation environment was limited to open-loop tests where a limited number of precomputed trajectories including their precomputed images could be used. Nevertheless the integration into Matlab/Simulink proved to be the right approach to take since this environment allowed easy debugging of inter-module communication and the analysis of effects which only occur in the interaction of modules. It also enabled the variation of architecture and configuration for the navigation system.

Figure 4.1 shows an exemplary result of the MiL tests. It shows the position and velocity error in MCMF coordinates. In this case the simulation starts at 1200 s of the descent and landing sequence. The reason for a start at 1200 s is that only from that point on are updates of the navigation solution possible. There are two reasons for this. First, due to the chosen time and trajectory the landing vehicle is over the dark side of the Moon. Therefore no craters can be detected and used for navigation. Secondly, the crater catalog for this simulation had to be manually prepared. The first region with known craters is visible at around 1200 s. Thus the simulation was initialized with expected initial navigation errors at that time.

The background of the plot is colored in order to provide information as to which image processing results can be used in the navigation filter for updating the states. The gray highlighted phases between 1200 s and 2200 s

are phases where a crater catalog is available and a navigation update from crater navigation is possible. In the phase from about 2300 s until the end of the trajectory the feature tracking results are used for updates in the filter. The blue regions denote the phases where this is the only update option (except for the altimeter and star tracker). The green phases denote phases where the 3D processing results are used for filter updates on top of the feature tracking results. The lander is from 1200 s to 2600 s in the descent orbit. The powered descent starts at 2600 s.

It can be seen that in the phases where an update is possible the navigation error as well as the covariances are reduced. This applies especially to the descent orbit phase (before 2600 s). Since in this setup a crater navigation was not possible later than 2200 s the feature tracking output was used to keep the error bounded. That works well for position and velocity until PDI. After PDI the velocity error grows due to dynamical effects and errors of the IMU. The position is also affected by this but its error is kept bounded by the feature tracking and a few updates for the position provided by the modules of the 3D chain. In the last green phase the 3D Matching can use LIDAR measurements. With the very well known topography of the landing site it provides very accurate measurements and decreases the navigation error for both position and velocity.

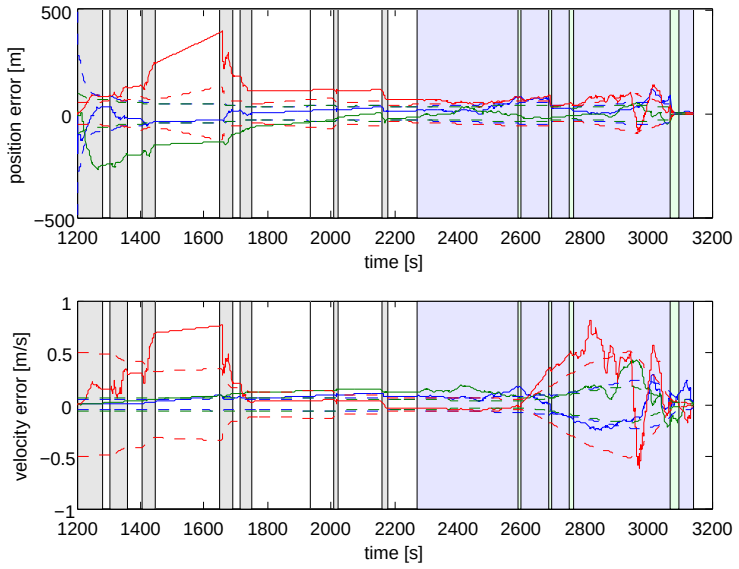


Figure 4.1: Results of the open-loop MiL simulation – navigation estimation error in MCMF coordinates: x – blue, y – green, z – red, dashed lines – covariances. The simulation starts at 1200 s of the descent and landing sequence where the first crater navigation results can be used. The gray highlighted phases denote phases where a crater navigation update is possible. The blue phases mark the phases where the feature tracking results can be used in the filter update. The green phases denote phases where the 3D processing and the feature tracking results can be used for navigation

4.1.2 Software-in-the-Loop Tests

As a further evolution the processing modules (see also Fig. 2.11) were embedded in DLR's Tasking Framework (see Sect. 3.9.3) which would be needed for the integration on an embedded system. The initial tests of the framework were also done in the Matlab/Simulink environment which is described in Sect. 3.8.3.1. The simulation module is depicted in Fig. 3.50

on Page 78. For the SiL tests, the cyan block in the picture doesn't contain the Matlab/Simulink model of the ATON modules but instead a single s-function that consists of a complete set of processing modules integrated in the Tasking Framework.

The main difference to the Matlab/Simulink integration is the presence of concurrency. The Tasking Framework allows the execution of parallel tasks and its scheduling is not bound to the Matlab/Simulink scheduler. Thus, the simulation is not stopped when a processing module is executed. This leads to slight degradation of the navigation state estimation compared to the MiL tests since other modules are not waiting until a result from a processing module is available. For instance, the the calculation of a 3D chain update takes several seconds; in the meantime the simulation has continued and the Navigation Filter has to update its state estimation with the older updates of the 3D chain when they arrive (see Fig. 4.2).

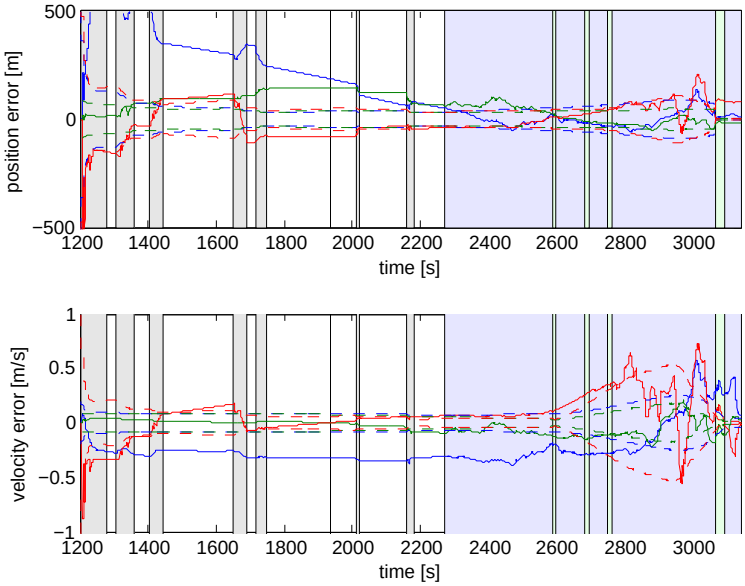


Figure 4.2: Results of the open-loop SiL simulation – navigation estimation error in MCMF coordinates: x – blue, y – green, z – red, dashed lines – covariances. The simulation starts at 1200 s of the descent and landing sequence where the first crater navigation results can be used. The gray highlighted phases denote phases where a crater navigation update is possible. The blue phases mark the phases where the feature tracking results can be used in the filter update. The green phases denote phases where the 3D processing and the feature tracking results can be used for navigation

4.1.3 Closed-Loop Model-in-the-Loop Tests

For proving function and performance in closed-loop operation the simulation was extended by models for vehicle dynamics as well as a guidance and control function. Furthermore the simulation was connected to the image simulation engine to compute the camera and LIDAR images based on the current true state vector which is influenced by the control actions.

Since the trajectory control is only active in powered descent the simulations were constrained to the time between 2600 s (PDI) and the landing at 3142 s. Initially, the states at PDI were initialized with the errors gained from open-loop simulations. That led to large deviations of the trajectory and led to situations where the 3D Matching did not work because the

camera was not looking at the expected terrain for which a matching was foreseen. In order to be able to still assess the behavior in closed-loop the initial errors were reduced allowing the modules to work in powered descent.

Figure 4.3 shows the navigation error of a closed-loop simulation of the powered descent. The errors and covariances are again given in MCMF coordinates. The background of the plot is again colored to indicate which image processing results can be used in the navigation filter. For the full powered descent the feature tracking results are used for updates in the filter (blue regions). The green phases denote phases where the 3D processing results should be used for filter updates. It can be seen that the updates of the 3D Maching provides good updates in particular in the second period from about 2740 s to 2770 s. In this period the error and the covariances can be reduced. Later the position and velocity errors increase again during the phase where only feature tracking is possible. In the final landing phase where the 3D Matching can use the precise measurements of the LIDAR only a few updates are done. Some of them have even a low quality. That leads to a large error in the velocity which subsequently accumulates in the position error.

The main reason for these degraded results is that the landing vehicle flies on a different trajectory with different attitudes, due to the controller's reaction to errors in the navigation solution. A result of this is that the camera and the LIDAR are pointing in different directions than in the nominal case where they cannot see the terrain for which reference data are available.

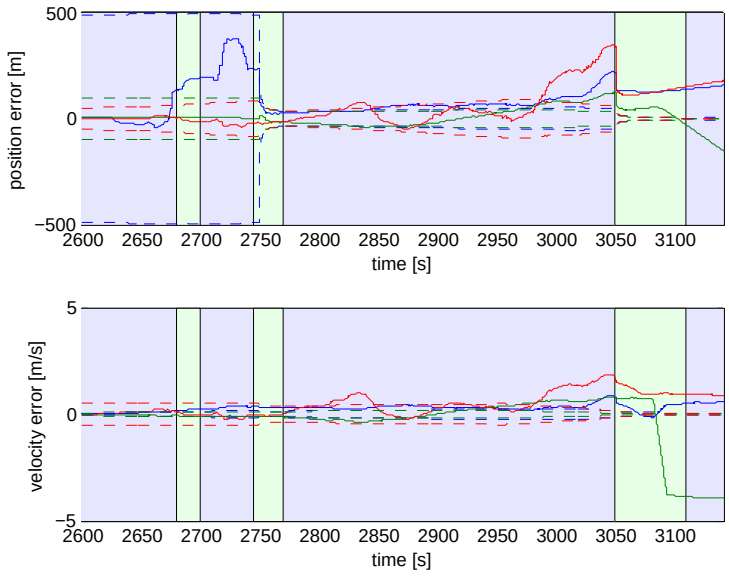


Figure 4.3: Results of the closed-loop MiL simulation – navigation estimation error in MCMF coordinates: x – blue, y – green, z – red, dashed lines – covariances. The simulation starts at the PDI at 2600 s of the descent and landing sequence. The blue highlighted phases mark the phases where the feature tracking results can be used in the filter update. The green phases denote phases where the 3D processing and the feature tracking results are used for navigation

Computing a single camera image took about 20 s. For this reason the closed-loop simulations became a lengthy exercise lasting several days for a single simulation of the powered descent with a length of about 600 s simulated time. Nevertheless, the effort to create the closed-loop envi-

ronment and to run the simulations was returned with results indicating how the control actions may influence the navigation function and performance. The conclusions are summarized in Sect. 4.1.4.

4.1.4 Conclusions for Model-in-the-Loop and Software-in-the-Loop Tests

After reviewing and analyzing the data of the MiL and SiL tests first conclusions have been made.

All simulations showed that the crater navigation has been working with the expected performance. This could only be shown for short intervals of the descent orbit because of two main reasons. First, the simulation uses a DEM from the Japanese Kaguya mission which has a ground resolution in the order of 10 m and an altitude resolution and error in the order of 50 m. For lower altitudes the error, quantization, and noise become dominant in simulated images which were generated by rendering a 3D scene. This leads to noisy and blurred craters in the simulated images. For the landing sequence there is no crater visible in the DEM because of its errors and limited resolution. Second, at that stage of the project the crater maps had to be created manually by picking craters in the DEM. For that reason the number of craters in the catalog was limited. However, the results show that the crater navigation provides an improvement of the positioning accuracy in the descent orbit. This allows a more accurate initiation of the powered descent thus enabling a more precise landing.

In all simulations the tracking of features worked as expected. The fusion of feature updates in the navigation filter must be carefully tuned to achieve the effect of bounding the expected growing velocity errors from IMU measurements and gravity field model errors.

The three modules of the 3D chain (Epipolar Geometry, Stereo Matching, 3D Matching) provided a navigation solution in short phases of the powered descent. In order to work, the dynamics of the terrain should be as large as possible. For this reason the processing chain was only enabled at times in the powered descent where it was expected that specific regions of the Moon are visible. These regions were selected because they offer high terrain altitude variation in a small area. This helped to reduce the effects of depth errors from the Stereo Matching on the 3D Matching. Additionally, larger dynamics of the terrain should lead to more unambiguous optimization solutions for the 3D Matching, which helps to increase the probability and accuracy of a solution from this module. Hence, low terrain dynamics lead to a reduced chance of success for the 3D Matching, which is a weakness of the 3D processing chain. However, at this point in the project it was the only method providing an absolute position measurement after the last crater navigation update until the utilization of the LIDAR measurements for 3D Matching.

In all simulations where the landing site was visible by the LIDAR, the 3D Matching performed well. It proved to be a very precise method for the final terrain-based absolute navigation shortly before landing.

During closed-loop simulations it turned out that deviations from the nominal trajectory led to a loss of absolute position matches. This could be overcome by enlarging the reference maps or the crater catalog. For the 3D

processing chain this option is limited by the fact that areas with large altitude variations are needed to get a good matching performance. For the crater navigation this would be in principle possible since crater maps can be acquired from geo-referenced images as they are provided by JAXA's Kaguya [78] and NASA's Lunar Reconnaissance Orbiter [54]. However, this cannot be proven in simulations since the DEM which has been used in simulations does not contain sufficiently small craters. Therefore, a proof cannot be provided in the MiL simulations yet. Reference maps for the 3D processing chain are rather larger in comparison to crater catalogs or BSM reference data, and the reference map size has a clear effect on the required computation time for the 3D Matching. Hence, a careful analysis is required for each mission in order to find a trade-off between the ability to deal with larger deviations from the trajectory and the available computational resources. Given an approximately correct state of the lander and appropriate 3D data, e.g. from a LIDAR, the 3D Matching showed its value as a final refinement step in order to achieve a high accuracy.

From the discussion above the following conclusions have been derived:

1. The reference maps or catalog data have to be enlarged to account for deviations in the closed-loop case.
2. Since it is difficult to include more maps for the 3D chain, alternative methods to gain absolute position information from image processing should be found. They should be able to work with noisy reference data at lower altitudes. One option is the idea of Binary Shadow Matching (see Sect. 3.4).
3. As a second alternative to the 3D chain, the crater navigation should be extended to the whole landing trajectory.
4. The 3D Matching, especially with LIDAR data, should only be used at a few strategically important points on the trajectory where high accuracies are required and an already accurate state estimate can be expected.

4.2 Hardware-in-the-Loop and Processor-in-the-Loop Tests

4.2.1 Open-Loop Hardware-in-the-Loop Tests

In Sect. 4.1 the ATON system was successfully demonstrated in a software-in-the-loop test where all sensor measurements were based on software simulation. In order to further develop the optical navigation components of the ATON system, the next step was to use real sensors in the loop. Consequently, the camera simulation was replaced by a real monocular camera.

Adding a real camera into the system creates a multitude of practical problems, caused by lens distortion, blur, sensor noise, limited dynamic range, unknown alignment between image reference system and camera reference system, hand-eye calibration, and so on. Solving these problems significantly increased the maturity of the optical navigation system for practical use.

Furthermore, the HiL testing cannot capture all conditions needed for all sensors as well as processing modules as amongst other reasons the use of selected downscaled scenes was necessary. This downscaling of the scene

works fine for cameras, but for LIDAR and other ranging sensors it does not work in the same way. The measurement error does not scale and thus does not allow for the inclusion of a LIDAR in the HiL simulation with the same setup as for the camera. In addition, some processing modules require reference maps for proper functioning (e.g. the 3D-Matching and the Shadow Matching). If those are not available these modules cannot be properly tested.

4.2.1.1 Test concept

In order to actually capture meaningful images, the visual environment, which is seen by the real camera during a lunar landing, must be simulated. For several reasons (for details see [47]) it is not feasible to replicate all the lunar surface visible during the mission. For the ATON project, the approach was to simulate selected subsections of the trajectory. In terms of optical navigation, there are three significant phases of the landing trajectory (see Fig. 3.49): the descent orbit, the powered descent, and the landing phase. These significantly differ in their geometric shape and in the applicability of optical navigation methods for them. Therefore, the goal of this milestone was to demonstrate a successful navigation during three sections of the lunar landing using a monocular camera sensor with hardware-in-the-loop simulation. Due to the aforementioned limitations, all other sensors remained to software-simulated.

This simulation took place at the TRON facility, situated at the DLR Institute of Space Systems in Bremen, see Fig. 4.4 and Sec. 4.2.1.2. It has been designed for providing scenes which represent exploration missions. TRON is a HiL test facility, with the purpose of supporting the development of optical navigation technology, which provides an environment that allows qualifying breadboards to Technology Readiness Level (TRL) 4, and qualifying flight models to TRL 7. Typical sensor hardware which can be tested in TRON are active and passive optical sensors like LIDARs and cameras.

During the actual test, the camera sensor was mounted at the robot's end effector, i.e. at the Tool Center Point (TCP), and moved along predefined

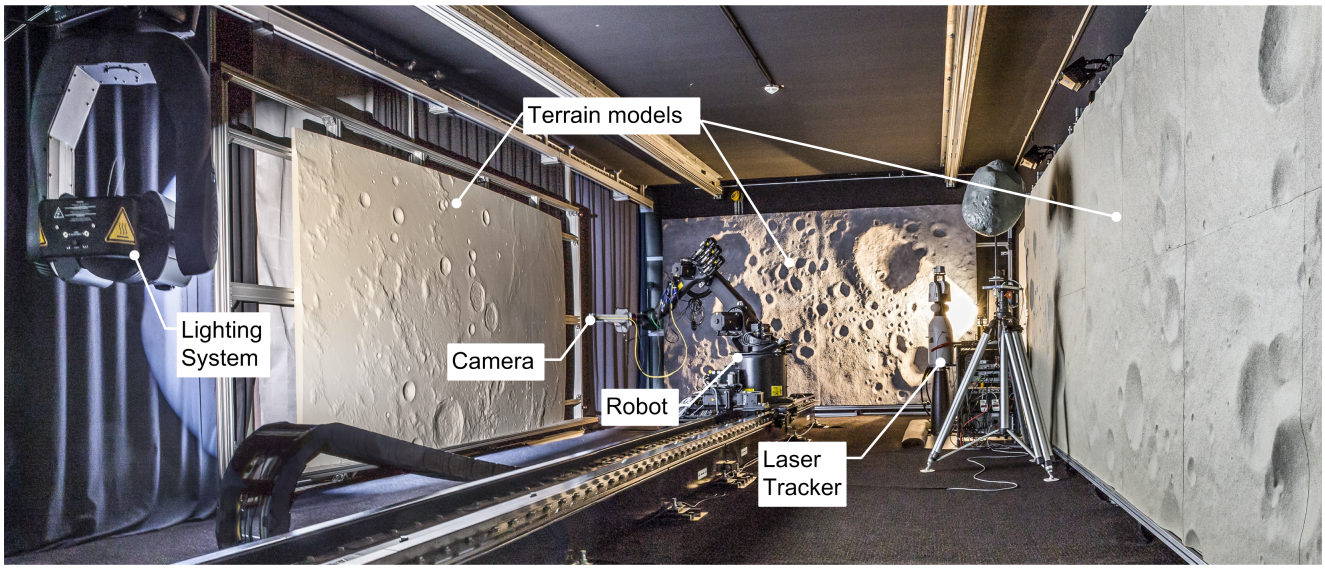


Figure 4.4: Simulation of the descent orbit phase of a lunar landing trajectory in TRON. The robot positions the optical sensor (in this case a camera) with respect to the illuminated terrain model, while the sensor is recording data. Simultaneously, the laser tracker measures the true pose of the sensor precisely with respect to the simulated Moon

trajectories. The movement was performed in a stop-motion manner, i.e. one position at a time at which all sensor data is recorded before the robot moves on to the next position. According to the trajectory, the sensor was positioned with respect to one of the three terrain models in TRON. At the same time a lamp, which simulates the solar illumination, was positioned in order to provide the expected illumination conditions. After a position was reached, an image was taken and fed into the ATON software. In the same step, a laser tracker measurement was taken for recording the true pose of the camera with respect to the terrain models.

4.2.1.2 Building blocks

The major building blocks of the lab are a robot on a rail for dynamic positioning of the sensor under test, a dynamic lighting system to illuminate the terrain models or other targets, laser metrology equipment for the recording of high precision ground truth data, and a dSPACE real-time system for test observation, control, and synchronization of ground truth and sensor data. For a better understanding of the test concept, the major building blocks are explained in the following.

Robot

The simulation of the dynamics is realized by a 7-DOF system comprising a 6-DOF KUKA KR 16 industrial robot on a 11 m linear rail which points along the long axis of the room. The possible payload of the robot's end effector at the TCP is 16 kg. The static repeatability of the robot is ±0.1 mm, the maximum TCP traverse velocity is 1.47 m/s. The robot is controlled either manually, by programs written in a robot script language, or by the dSPACE real-time system. The dSPACE system can also be used for potential real world simulations of the spacecraft. By default, the sensor to be qualified will be installed at the TCP. Should the candidate technology exceed the payload mass of the TCP, the robot's arm and base allow additional hardware to be placed, up to a total payload of 40 kg.

Terrain models

The TRON laboratory room was designed for an installation of suitable 3D

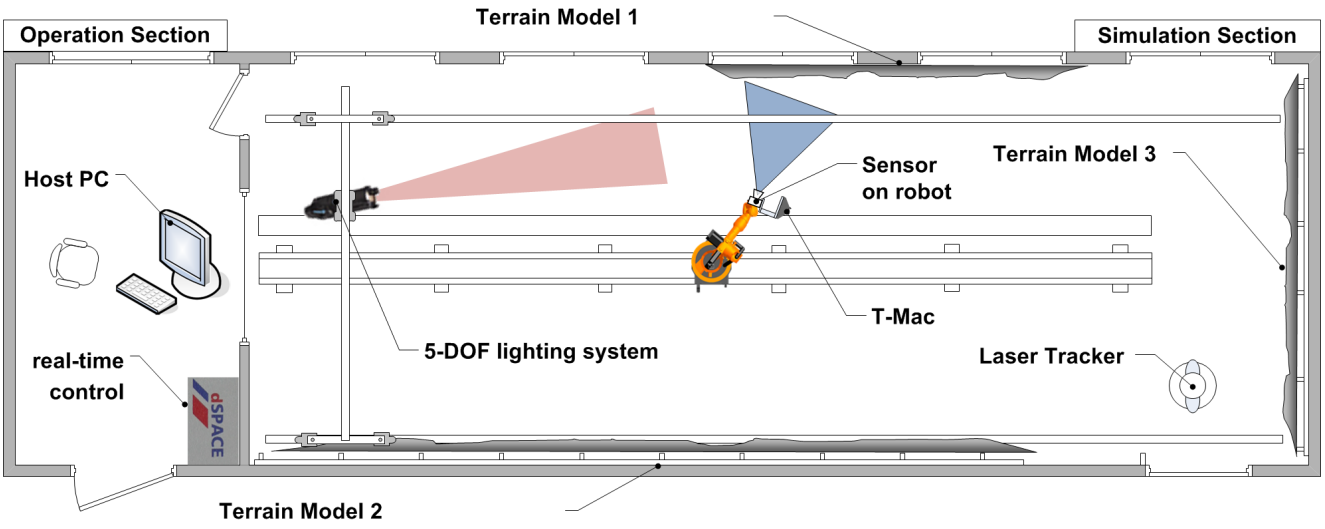


Figure 4.5: Schematic overview of TRON's building blocks

terrain models on three walls. Hence, it was possible to achieve a coexistent simulation capability of multiple relevant parts of a typical mission profile of a full lunar landing trajectory. Furthermore, it was planned to illuminate the 3D models with a suitable lighting system for achieving high quality shadows in real-time. Three terrain models have been designed, manufactured and installed and are explained in the following. The naming of the models is according to Fig. 4.5.

Terrain model 1, as shown in Fig. 4.6, was installed along the windowed wall in TRON (see Fig. 4.5). The model has the size of 3.92 m × 1.96 m and represents a part of the Moon scaled to 1:125000 (see Fig. 4.6). The terrain dynamics, i.e. the range between the lowest and the highest part, is about 20 cm. The model has been milled to an accuracy of 1 mm. The model reference data has been derived from Kaguya DEMs. Using this model, high-altitude orbits like the parking orbit as well as the first part of the DO can be simulated. This model is truly Cartesian, therefore including the natural curvature of the lunar surface. Within the ATON project it was used for simulating parts of the DO, see also Sect. 4.2.1.4.

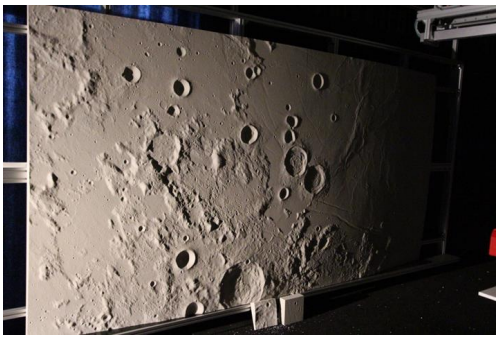


Figure 4.6: Photo of terrain model 1 (1:125000 scale model of the Moon created from Kaguya 3D data)

The terrain model 2, as shown in Fig. 4.7, was installed at the long windowless side of the room section and it measures 9.80 m × 1.96 m. The terrain dynamics is about 6.2 cm. Like the terrain model 1 it was milled with an accuracy of 1 mm. Its reference data was software-generated for being representative of the lunar surface. Due to the self-similarity of crater size distribution and its definition in the cross-range, down-range, altitude system, it can represent a lunar surface at different scales. As an exemplary use case the robot's TCP was moved along the entire terrain model 2 with a variable distance between approximately 0.2 m and 3 m. Considering this terrain model at a scale of 1:10000, the SC position could be simulated over a downrange distance of 100 km and altitudes between 2 km and 30 km.

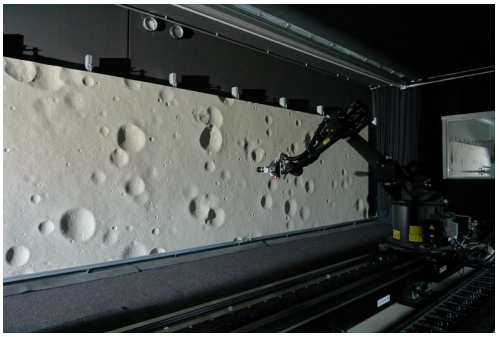


Figure 4.7: Photo of terrain model 2

Terrain model 3, as shown in Fig. 4.8, was installed on the front wall of TRON. Its size is about 4.20 m × 2.20 m, and the terrain dynamics is about 26 cm. The model reference data was obtained entirely by DLR via a process that started with hand-modeling of a potential lunar terrain and ended with the 3D-scanning and post-processing as described in [51]. The model was then manufactured in two steps. At first, a coarse milling step obtained the rough terrain structure. Afterwards, a finishing surface layer was applied manually. Due to the hand-made finishing, no manufacture marks such as milling lines are visible, leaving the model with a practically infinite resolution. Again the self-similarity of the model and the Moon can be exploited by applying a different scale to this model. This landing site model is not only representative of the lunar surface, but also for many asteroid surfaces. It is destined for being used for the simulation of the last phase of the landing. In this way, the terrain-relative navigation with respect to the landing site and the evaluation for safe areas can be tested with a HiL test. Combining this model with low-scale factors makes it a useful sensor target for 3D imaging sensors. For the ATON project, the model is considered to have a scale factor of 1:100.

The TCP can be moved from a distance of 11 m to a distance of 1 m towards the terrain model 3, and at the same time with a radius of ≈1 m perpendicular to the rail. In this way, an approach from an altitude of

1100 m down to 100 m can be realized along with a possible simulated lateral movement of the SC of 200 mm.

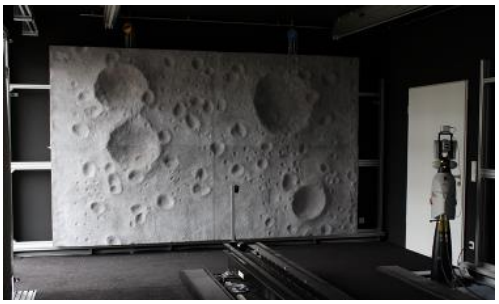


Figure 4.8: Photo of terrain model 3

Space background simulation

Space-like background is obtained by the use of dark and anti-reflective materials in the laboratory room. This entails moving black curtains which can close all windows of the simulations section in order to exclude any light coming from outside the lab. Any ambient lighting originating from internal light sources is avoided by black paint on the walls and the ceiling, with additional black curtains that cover the walls and with black carpet on the floor.

Lighting system

The lighting system, as shown in Fig. 4.9, contains a 3-DOF gantry with a light source installed on it. The light source is the zoom profile spotlight ADB WARP, using the Hydrargyrum medium-arc iodide (HMI) technique to achieve an efficient light production out of its 575 W electrical power. The lamp is optimized for uniform lighting, and the color temperature is 6000 K. The lamp can be rotated about two axes, also various other parameters such as zoom, the focus, gobos or the mechanical dimmer can be dynamically changed. In combination with the gantry the 5-DOF system provides variable solar irradiation angles within the whole simulation section.

Figure 4.9: Lamp installed on gantry's payload platform

Laser metrology equipment

In TRON, a laser tracker system is used as the main tool for precisely measuring position and pose of lab elements. It is used for tasks such as the alignment between the building blocks, dynamic position iteration and the recording of ground truth data. In the ATON project the following components were used:

- The **AT901-MR laser tracker system** (see Fig. 4.10a) is the central device of the laser metrology equipment. Depending on the tool the laser tracker is combined with, the tool's 3D position or 6D pose is measured with respect to the Laser Tracker reference system (LTRF).
- Single **Reflectors** (see Fig. 4.10d) are used for measuring 3D points in the LTRF. The reference is the reflector's center. In the ATON project the reflectors were used to determine the pose of the Terrain Model reference system (TMRF) with respect to the LTRF.
- The **T-Mac** (see Fig. 4.10b) comprises a set of LEDs and a reflector. The positions of both are measured by the laser tracker and processed, resulting in the 6D pose of the T-Mac given in the LTRF. In the ATON project the T-Mac was used for determining the pose of the camera with respect to the LTRF.
- The **T-Scan** (see Fig. 4.10c) is used for digitizing the geometry of a surface via a laser line scanner. Like the T-Mac the T-Scan is equipped with reflectors and LEDs for determination of its pose within the LTRF. Both the surface samples and the pose of the T-Scan result in a point cloud of the surface within the LTRF. In the ATON project the T-SCAN was used to acquire DEM of the lunar terrain models in TRON.

Table 4.1: TRON laser tracker system performance

Device	Position Accuracy (MPE)	Attitude Accuracy ($\frac{1}{2}$ MPE)	Working Distance
3D position	$\pm 15\,\mu m + \pm 6\,\mu m/m$	NA	1.5 m - 25 m
6D pose	$\pm 15\,\mu m + \pm 6\,\mu m/m$	± 0.01deg	1.5 m - 9 m

The accuracy of the laser tracker system is shown in Tab. 4.1.

4.2.1.3 Ground Truth

In order to measure the performance of the navigation system its navigation solution must be compared to the true state. The navigation system delivers the pose of the spacecraft reference system (SCRF) with respect to the MCMF reference system. The ground truth also contains this pose, but from independent measurements. In TRON, the camera is positioned with respect to a terrain model which represents a part of the Moon. Depending on the terrain model it is either an actual scaled section of the lunar surface, or a Moon-typical artificially generated terrain, associated with some appropriate scale factor (cf. Sec. 4.2.1.2). Determining this pose requires the knowledge of the transformation which leads from the true scale Moon, i.e. MCMF coordinates, to the scaled terrain model in the lab with TMRF coordinates and then to the camera and to the SCRF:

$$\text{MCMF} \Leftrightarrow \text{TMRF} \Leftrightarrow \text{SCRF} \tag{4.1}$$

Since the ground truth cannot be measured directly, a set of tools such as reflectors, the T-Mac, and a laser scanner are used in combination with

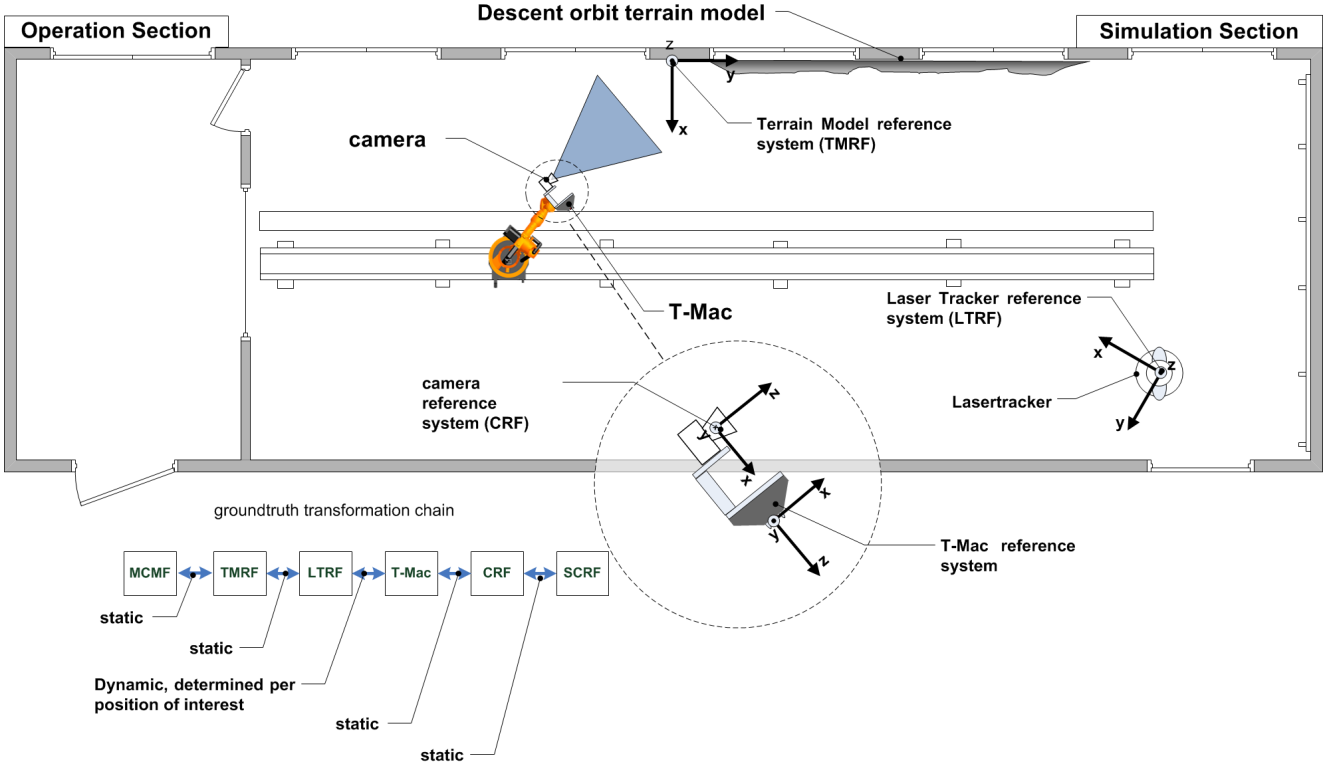


Figure 4.11: Overview of the complete TRON ground truth transformation chain

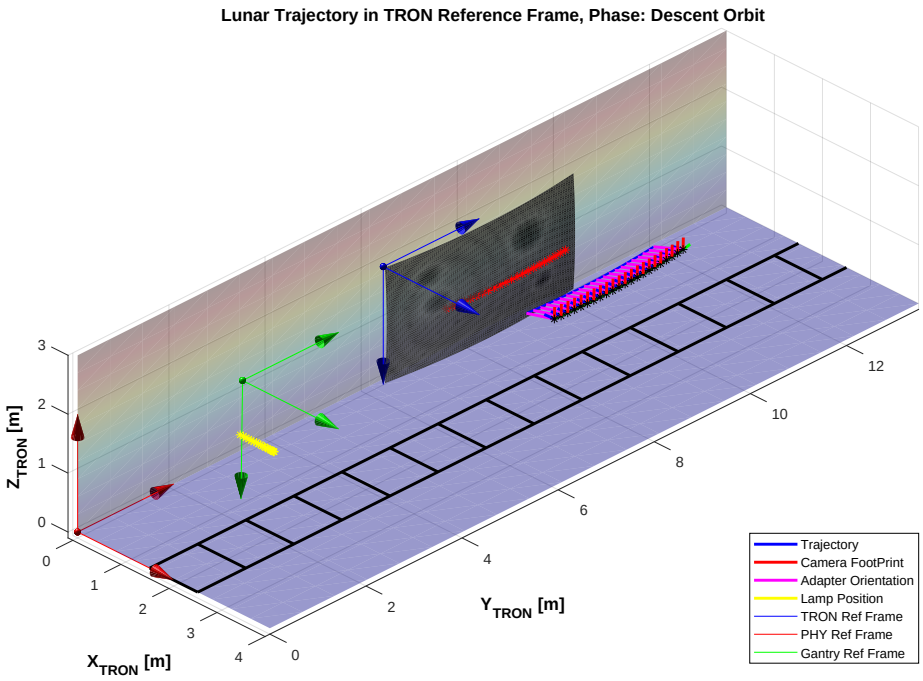


Figure 4.12: Illustration of the section of the descent orbit trajectory simulated in TRON

(a) AT901-MR laser tracker

(b) T-Mac for 6D pose measurements

(c) T-Scan for point cloud measurements

(d) Laser reflector installed on reflector holder

Figure 4.10: Laser tracker tools used for the ATON project

methods such as hand-eye calibration and point cloud matching. In this process, several intermediate reference systems are introduced. Consequently, 4.1 expands to 4.2 (see also Fig. 4.11). All but one of the transformations are static and have to be determined only once before the test. Whenever the robot moves, the transformation between the laser tracker and the T-Mac is subject to change and is measured directly with the laser tracker.

In preparation for the test, the complete set of static transformations was determined:

$$\text{MCMF} \Leftrightarrow \text{TMRF} \Leftrightarrow \text{LTRF} \Leftrightarrow \text{T-Mac} \Leftrightarrow \text{CRS} \Leftrightarrow \text{SCRF} \tag{4.2}$$

For details, please refer to [48].

4.2.1.4 Trajectory

The trajectories determined during the ATON project describe the spacecraft's pose with respect to the Moon throughout the complete lunar landing mission. For the HiL simulation, only three sections of one of the trajectories were simulated (see Sect. 4.2.1.1). The overall approach tries to position the camera and the lamp relative to the terrain model in TRON according to the given poses of the SC and the Sun from the lunar landing trajectory. For each pose selected from the trajectory the robot, the gantry and the lamp were commanded accordingly. On the basis of the alignment performed in Sect. 4.2.1.3, these commands could be successfully determined, please see [76] for more information. Figures 4.12, 4.13, and 4.14 illustrate all three sections of trajectory.

4.2.1.5 Position refinement

Positioning the camera in TRON is realized via the 6-DOF robotic arm, which is installed on an 11 m rail. Due to effects such as non-linearity of the

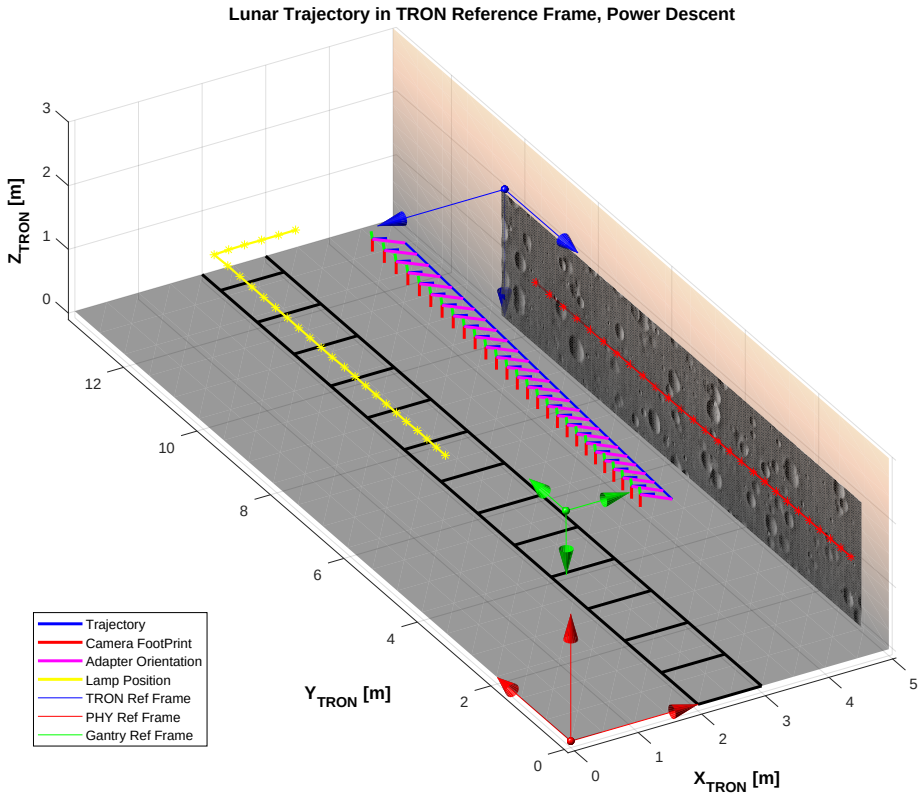


Figure 4.13: Illustration of the section of the powered descent trajectory simulated in TRON

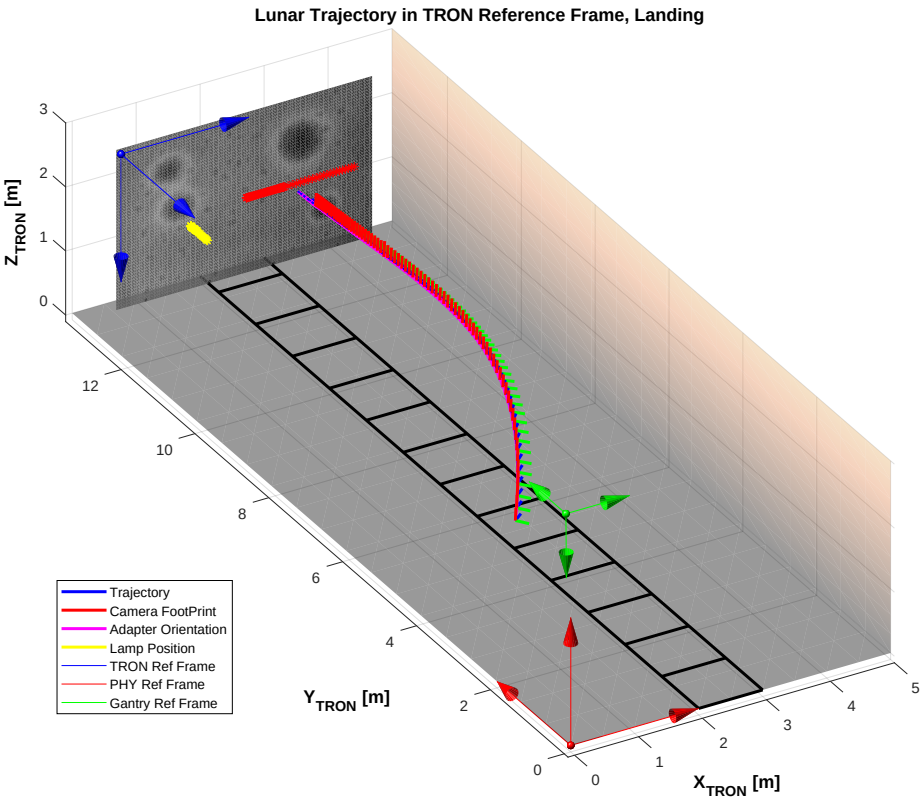


Figure 4.14: Illustration of the section of the final landing trajectory simulated in TRON

rail, errors are introduced into the robot's position solution. The resulting displacement is in the order of 1 cm. However, due to the efforts with regard to the ground truth in Sect. 4.2.1.3, the pose of the T-Mac with respect to the terrain model is known with sub-millimeter accuracy. Using hand-eye calibration methods, the transformation between the robot's TCP and the T-Mac can be determined with an accuracy of less than one mm and less than 0.01 deg. As the transformation between the T-Mac and the camera is also known with high precision, it is also possible to increase the performance of the robot in terms of positioning the camera with respect to the terrain model. The procedure for this is as follows:

1. Command the robot with the initial trajectory and measure it with the laser tracker in LTRF,
2. Transform the measured poses into the robot coordinate frame,
3. Determine the difference between the reference pose and the achieved pose,
4. If the error is above the required threshold, use the difference to generate a corrected trajectory, and
5. Return to step 1 and use the corrected trajectory instead of the initial one.

After two iterations, the maximum discrepancy between the reference and measured trajectories decreased from 8 mm and 2.5 deg to less than 0.3 mm and 0.3 deg, respectively. Further iterations have been performed, but no significant improvements in the magnitude of the error were observed. The results are shown in Fig. 4.15.

4.2.1.6 Results

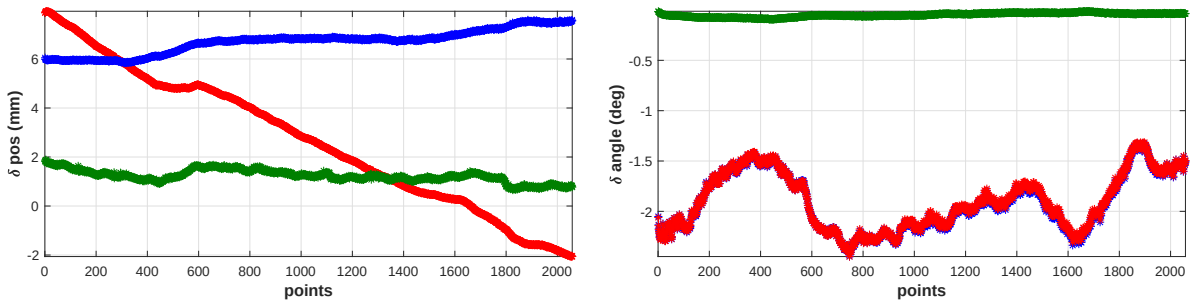
Module tests

Prior to tests with the integrated ATON system, the images generated in TRON have been used in the development of the image processing modules. This section outlines for one of the modules the results obtained during this process.

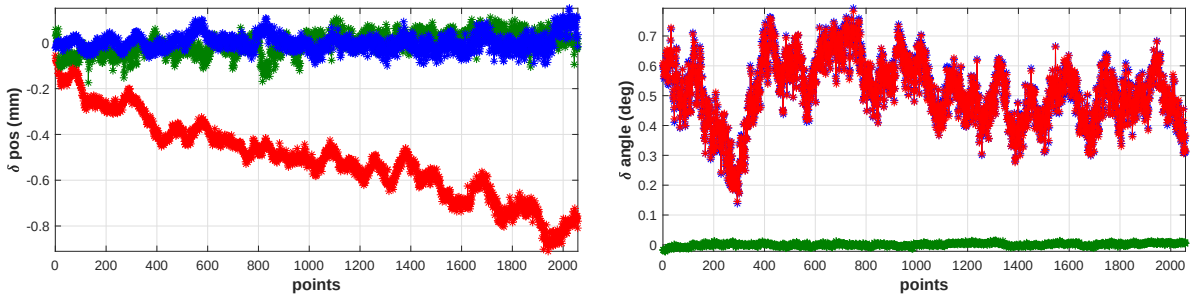
To test the Crater Navigation module, the terrain model 1 of TRON was employed. Since it is based on a DEM generated from data of the Kaguya mission, it represents a realistic sub-scale model of the Moon. For the lunar DEM, a crater catalog was created. As described in the previous sections, the trajectory of the descent orbit was divided into several segments. Each segment was simulated with the same terrain model and the segments were realized as consecutive passes over the model with slightly altered illumination. The catalog was transformed for each pass in the proper MCMF location. With this setting, the Crater Navigation module was tested for the part of the descent orbit which has sufficient illumination (in this case 1200 s after the beginning of the complete landing sequence). The image acquisition rate was set to 1 Hz.

Figure 4.16 shows two examples of images from this sequence. In the images, the detected craters are marked by turquoise ellipses. After obtaining a camera pose from detected and matched craters, the catalog is projected onto the image. All craters of the catalog are marked by pink crosses. Overlapping ellipses and crosses then indicate a match between a detected crater and a crater in the catalog.

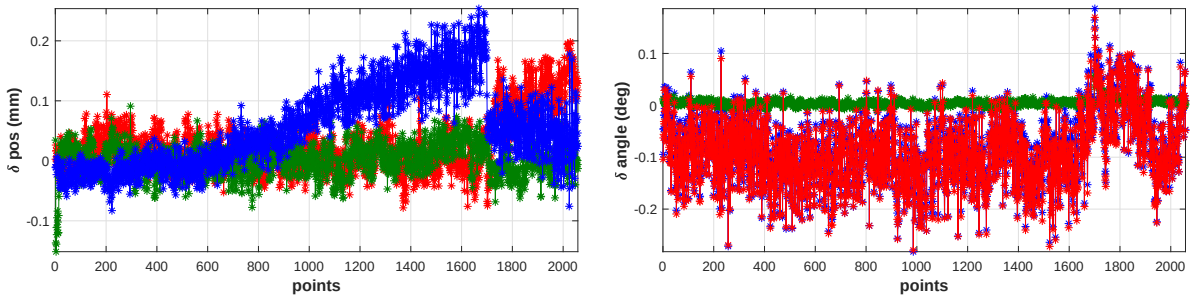
For each image the Crater Navigation module was applied and the resulting position estimates were compared to the ground truth provided by the



(a) First run: difference between the reference trajectory and the measured trajectory in the robot's reference system



(b) Second run: difference between the reference trajectory and the corrected trajectory (1st iteration) in the robot's reference system



(c) Third run: difference between the reference trajectory and the corrected trajectory (2nd iteration) in the robot's reference system

Figure 4.15: Differences between the initial trajectory and different corrected trajectories

TRON laboratory. Figure 4.17 shows the magnitude of the positioning error scaled by the line-of-sight distance of the camera. It can be seen that the error is within 1 % for the majority of images. Figure 4.18 provides a histogram plot of the error in camera coordinates, again scaled by the line-of-sight distance again. It can be observed that the lateral axes (x, y) have a slightly larger dispersion than in the viewing direction. Furthermore,

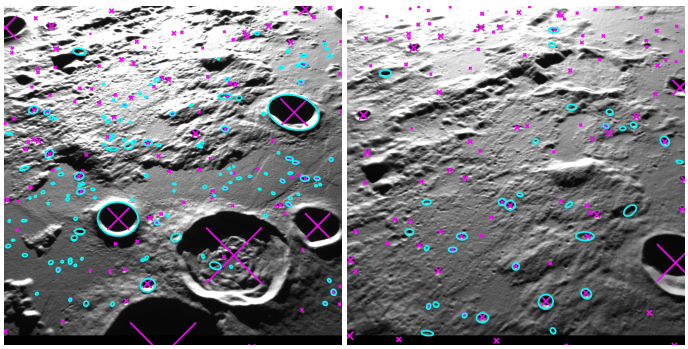


Figure 4.16: Crater navigation during lunar landing simulation in TRON. Turquoise ellipses: detected craters, pink crosses: craters in database. Overlapping symbols indicate a match between detected craters and database entries used for navigation

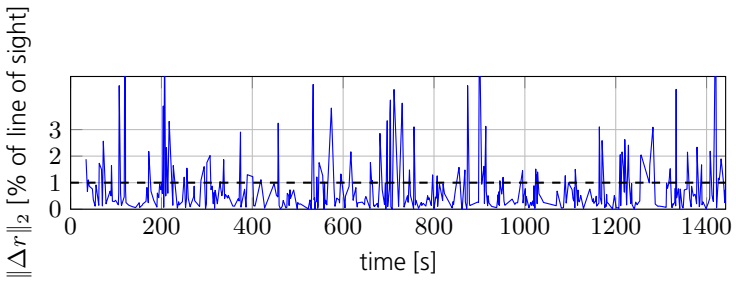


Figure 4.17: Crater navigation relative position errors for the descent orbit simulation in TRON using terrain model 1

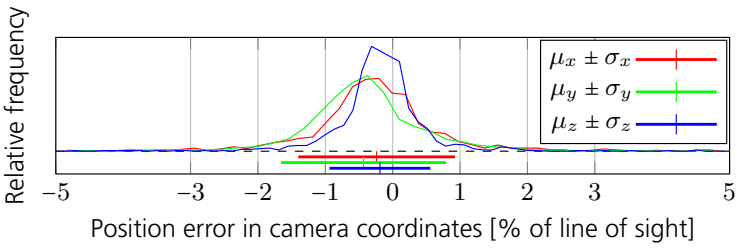


Figure 4.18: Crater navigation signed position error histogram for the descent orbit simulation in TRON using terrain model 1

it can be seen that the mean value for all errors is slightly shifted from zero. This indicates a small systematic error. In comparison to other results shown in Sec. 3.1.4, where the mean error in the lateral axes was zero, this offset can be attributed to a systematic error in the laboratory setup. The main cause, however, is that the catalog was created from a source DEM of the Moon that is given in MCMF coordinates. The physical lab model, which is made of four tiles, deviates slightly from this DEM by up to 5 mm. With a scaling of 1:125000, this translated to an error of up to 625 m. For a line of sight of 100 km, which accounts for a relative error of up to 0.625 %. Since the deviation of the lab DEM from the lunar DEM is not evenly distributed, the effect on the Crater Navigation performance is not constant.

Similarly the Feature Tracker module was tested with terrain models 2 and 3 in a simulation of the powered descent. To provide an impression Fig. 4.19 shows two example images with tracked features. The results of the Feature Tracker were used in the system simulation to feed the Navigation Filter module. Although the Epipolar Geometry module and the Stereo Matching module could not be used for the navigation processing, the recorded images have been used to verify the function of these modules with real camera images.

Integrated Tests

As discussed previously, only short sections of the whole landing sequence can be tested in the lab. Furthermore, creating reference data for the Shadow Matching as well as for the 3D-Matching was not possible as appropriate DEMs of the terrain models 2 and 3 in TRON were not available at the time of testing. This resulted in the following setup for the HiL test in TRON:

- The Crater Navigation module was tested with the Navigation Filter in phase 1 of the landing sequence (see schedule of modules and sensors in Fig. 4.21).

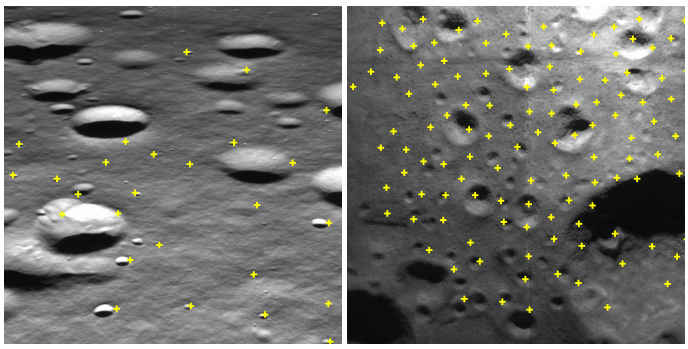


Figure 4.19: Using the Feature Tracker within the TRON test bed. The pictures show the powered descent (left) and the final landing phase (right) with features used for relative navigation

- It was not possible to test the 3D-Matching processing chain as well as the Shadow Matching module.
- During the powered descent, only the Feature Tracker module outputs were fed into the Navigation Filter together with simulated data from the LA and the STR. This applied to the early powered descent (phase 2 simulated with terrain model 2) and the final landing (phase 3 simulated with terrain model 3).
- LA measurements were simulated with a simplified model. It was assumed that the terrain models 2 and 3 are planar. The altitude measurement was the distance along the line of sight of the simulated laser to the plane of the average altitude of the terrain models.
- The camera was properly calibrated and all images were undistorted before feeding them into the processing chain.

Although the above mentioned points limited the test scope, the tests provided a good indication of new errors and effects that are introduced when changing from simulated to real images. Figure 4.20 shows the results of one of the test runs in TRON. It represents a short section of the early phase of the powered descent. The time again refers to the reference scenario where the PDI is at 2600 s and the landing at 3142 s. The test was performed using terrain model 2.

Since the navigation filter is updated only by outputs of the Feature Tracking, LA measurements and STR measurements, corrections of the position were not expected. For this short fraction of the powered descent, also no real impact on the velocity error can be seen.

4.2.2 Closed-Loop Hardware-In-The-Loop Tests

When changing from open-loop to closed-loop simulation in TRON, a few changes were necessary. The camera simulation module was replaced in the closed-loop simulation by a software module acquiring an image as shown in Fig. 4.21. Hence, in addition to the transformations needed for the GNC module, the transformation to the laboratory reference frame also had to be computed for every time step.

The settings of the laboratory, including the transformation parameters of the trajectory arcs to the laboratory frame, have been re-used. The main difference was that it was not possible to apply the positioning refinement as described in Sect. 4.2.1.5 to the whole trajectory at once, but it was instead necessary to execute it for each single position. Hence, for each

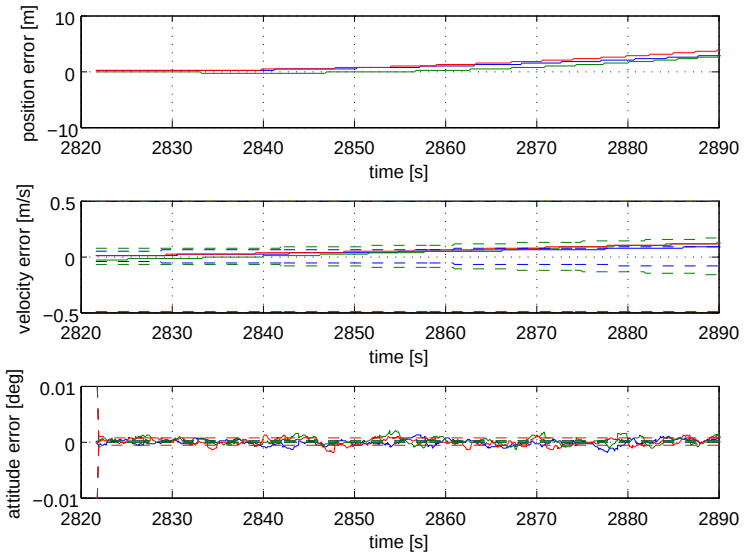


Figure 4.20: Results of an open-loop test in TRON. The simulated phase is a small fraction of the powered descent in high altitudes shortly after the PDI. During the PD, no absolute position updates were used. The initial error for the position was intentionally set to zero

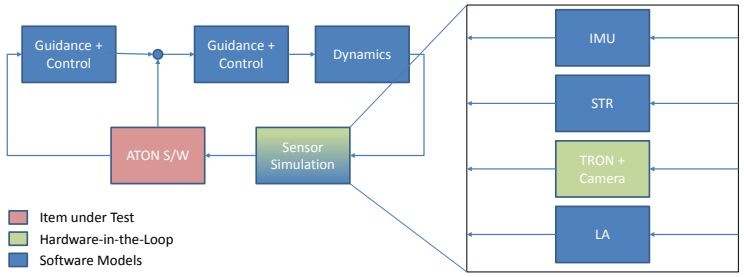


Figure 4.21: Block diagram of HiL closed-loop setup in TRON

single time step, when a camera image was taken, the dynamics simulation provided the current vehicle state. This was transformed to the laboratory coordinate frame and the robot was commanded to this pose. The laser tracker measured the current position from which corrections for the robot's movement were computed. This position refinement was done in at least three iterations before the next camera image was taken. Therefore, the simulation loop stopped for the time duration needed by TRON to put the robot in the required accurate pose. After the image was delivered to the ATON system, the processing and simulation loop was continued.

Using this setup, closed-loop HiL tests were conducted for two powered descent phases. The phase during the descent orbit was omitted since there is no closed-loop control involved. The results of the closed-loop tests do not differ from the open-loop simulations, which indicates that no significant effect was introduced by closing the loop. Due to the limitations of the lab tests in TRON, the 3D-Matching processing chain was not included in the test. For the MiL tests, the main difference between open-loop and closed-loop occurred due to the results of the 3D-Matching processing chain. Thus, similarity of the results can be expected.

4.2.3 Processor-in-the-Loop Tests

As a preliminary step to the planned flight tests, the ATON software was implemented on an embedded system. In a first step, the simulated data from the MiL simulations were fed into the navigation software to prove its function and performance on the embedded system in a real-time environment. Later, the same setup was used to replay recorded flight data in order to analyze different software settings and processing parameters. For both steps a real-time capable log player was developed. The architecture of the setup is described in Sect. 3.8.3.2.

The results of the PiL tests with the lunar scenarios were virtually identical to the SiL tests in the Matlab/Simulink environment with the integrated ATON software in the Tasking Framework. This can be explained by the fact that the visual flight computer, which was selected for the flight experiments, has a similar computing performance compared to the computer on which the Matlab/Simulink simulations were conducted. Furthermore, the sensor input for the log player was converted from the simulation environment. Hence, no changes in performance were expected and none were observed.

4.2.4 Conclusions for Hardware-in-the-Loop and Processor-in-the-Loop Tests

As for the MiL and SiL tests, first conclusions can be drawn from the HiL and PiL tests.

In TRON realistic scenes of exploration missions are simulated by using a robot for positioning active or passive optical sensors, terrain models and an illumination system. For ATON the navigation system's monocular camera was put into a TRON setup simulating the lunar environment. The lunar landing trajectory was divided into three sections, each one applied to one terrain model. For the descent orbit terrain model 1 was used applying a scale of 1:125000, for the powered descent terrain model 2 was used applying a scale of 1:10000 and for the landing section terrain model 3 was used with a scale of 1:100.

Two optical navigation methods were in the focus of ATON's HiL campaign. The crater navigation method was tested on terrain model 1. Beforehand the necessary crater database was obtained by processing DEM data of that model. Terrain models 2 and 3 served as test environments for the feature tracking method. Measurements based on both methods were fed into the navigation filter module, which can therefore be considered as the third module under test.

A total number of 1442 images were obtained for the descent orbit segment and processed by the crater navigation. By comparing its measurement outputs with the ground truth obtained by TRON's laser metrology equipment the accuracy was determined. As can be seen in Fig. 4.18, for the majority of the images the positioning error scaled by the line-of-sight distance of the camera is in the order of 1 %. With this result the crater navigation was within the expected performance. A total number of 3642 images were obtained for testing the feature tracking method during the powered descent and the landing simulation. The performance here was also as expected.

The images and data generated in TRON were very valuable for the development of ATON. The image processing modules in particular have been improved significantly by using real images, as effects were present which cannot be superficially covered by simulation.

In summary, after setting up the tests, calibrating the camera and robot, and debugging of interface and timing issues, the results of the simulation periods did not show deviations or other unexpected behavior. This indicates that the introduction of the real camera sensor into the processing chain is properly handled by all processing modules as they did not create new errors. Similar behavior was observed for the closed-loop tests which covered the powered descent and the landing part. Furthermore during the PiL it was shown that the ATON navigation software can run in real-time when processing the inputs of the complete sensor suite.

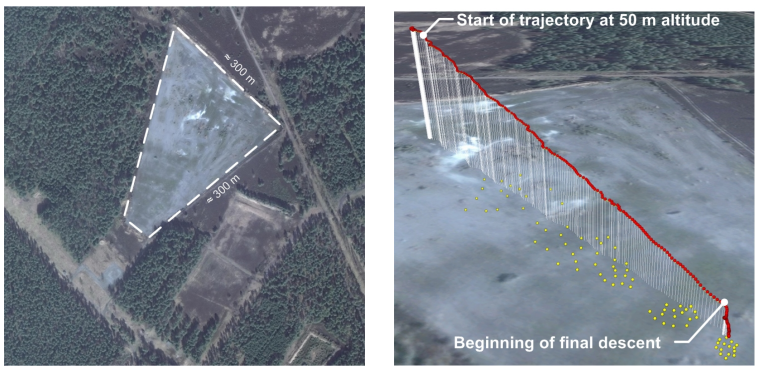
With the successful HiL and PiL tests the ATON navigation system was ready to take on the next development steps. One step is to integrate the computing hardware from the PiL with the real sensor hardware. A second step is to increase the number of sensors with respect to the HiL by also integrating an IMU, two altimeters and a second camera. Furthermore for proving real-time capability the ATON system is to be integrated closed-loop into a system with dynamics more closely resembling those of an actual lunar landing. The implementation of these steps is described in Sect. 4.3.

4.3 Flight Tests

Before conducting the flight tests, several other development steps had to be performed. First, the hardware had to be integrated, and interfaces and software drivers had to be developed, implemented, integrated and tested. Furthermore, the development included the design and production of targets representing craters. Also the design, implementation and verification of generating ground truth data and the mapping of the targets was carried out.

As pointed out earlier in this report, on-ground testing of GNC systems does not allow complete verification of the item-under-test in a single test, as the testing environment is different from the real-world environment and also differs from the trajectory expected in operational conditions. The same applies to the flight tests. It is obvious that the illumination conditions on the Moon cannot be recreated easily on Earth. For logistics reasons and labor intensity, the lunar landscape cannot be replicated on areas large enough for flight tests. Finally, the flight dynamics of a helicopter are different from those of a lunar landing vehicle.

Preliminary tests were performed by mounting the entire experimental setup on a small carriage and driving it around on the ground, in order to stimulate the sensors, firstly to obtain useful measurement data and then to verify the alignment of the setup. Later, flight tests were conducted using an unmanned helicopter, and the focus was on recording a consistent set of flight data from all available sensors. This was followed by a second flight campaign were the ATON navigation system was tested in open-loop. The final test campaign was concluded in March 2017. In these tests the ATON system was used as the primary navigation system for the autonomous flight of the unmanned helicopter. The results of this most recent test campaign are presented in the following sections.



(a) Overview of flight test area
Image background: Google Earth

(b) Trajectory of one test flight (red) and crater center positions (yellow)
Image background: Google Earth

Figure 4.22: Overview of test area and trajectory

4.3.1 Flight Test Setup

The overall test concept was to fly a navigation sensor suite along a predefined trajectory over a field of craters which had been mapped into an Earth-fixed reference frame. During flight, data from the sensor suite and ground truth were acquired simultaneously. The flight's objective was to demonstrate the real-time closed-loop operation of the optically augmented ATON navigation system in an exploration mission scenario. More information on previous tests of this platform can be found in [1].

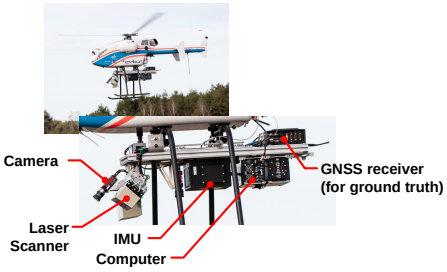
4.3.1.1 Trajectory and flight apparatus

The test campaign took place near Braunschweig, Germany, at a test site offering a strip of land and volume of restricted airspace suitable for flying unmanned vehicles over an area of about 300 m × 300 m (Fig. 4.22a). The function of transporting the navigation payload was performed by an unmanned SwissDrones (SDO 50 V2) helicopter (Fig. 4.23a). This platform is capable of autonomous, assisted and remote-controlled flight and it offers a carrying capacity of approximately 50 kg, which includes fuel and the experimental payload.

All sensors were integrated on a single rigid platform, facilitating easier mutual alignment. The devices used are marked in the image of the experimental payload in Fig. 4.23a. A tactical-grade IMU (iMAR iTraceRT-F400-Q-E, specifications on Table 4.2) was used for acquiring velocity and angle increments. It operated at 400 Hz.

Capturing of images was performed by two monocular, monochromatic cameras (AVT Prosilica GT1380). Having been installed in a forward-looking and down-looking configuration, their resolution was set to 1024 px × 1024 px. For measuring the altitude of the platform similar to the description of the lander in Fig. 2.5, a laser scanner (SICK LD-MRS) is used. The laser scanner has been configured to have only a small field of view to emulate a laser altimeter. The star tracker measurements could not be acquired during daylight. Therefore, they have been emulated by the reference navigation system which was also used to provide ground truth navigation.

Considering the experience of earlier activities with the ATON system, a position accuracy in the order of low one-digit percent of (camera) line-



(a) Camera, IMU, laser scanner, on-board computing and ground truth hardware installed on helicopter during flight

(b) Craters after preparation and ready for testing

Figure 4.23: Setup of payload hardware and craters

Table 4.2: IMU (1σ) specifications

	Gyroscope	Accelerometer
Sensor range	± 450 deg/s	± 5 g
Axis misalignment	0.5 mrad	0.5 mrad
Angle/vel. random walk	0.1 deg/$\sqrt{\text{h}}$	$50\,\mu$g/$\sqrt{\text{Hz}}$
Bias repeatability	0.75 deg/h	2 mg
Scale-factor repeatability	300 ppm	1500 ppm

of-sight range was assumed as a likely upper bound, as the detection algorithm's performance is slightly impacted when operating with the artificial crater targets instead of real craters. Given the flight trajectory followed (Fig. 4.22b), this translates to a ground truth accuracy requirement at centimeter level. Therefore, the helicopter payload was equipped with a high-grade GNSS receiver NovAtel Propak6. This device uses both L1 and L2 frequencies and the German precise satellite positioning service, Satellitenpositionierungsdienst der deutschen Landesvermessung (SAPOS). This service relies on a network of reference stations with precisely known positions that determines corrective data for all visible GPS satellites. Two GNSS antennas were used, allowing the receiver to also determine heading and pitch in the North-East-Down reference system. The Propak6 output has the following 1σ accuracies: about 0.03 m in position, about 0.4 deg in heading and pitch, and about 0.03 m/s in velocity.

About half of the available terrain in Fig. 4.22a was used for the flight trajectory. The remainder was reserved as safety perimeter, ground station and test crew area. The reference flight trajectory was defined as a linear path, stretching from north-east to south-west for about 200 m, and from an initial altitude of 50 m down to 5 m. From that altitude, the helicopter performed a vertical descent down to 1 m above ground. Fig. 4.22b illustrates this profile.

Since craters are required for the Crater Navigation module to work, a pattern of planar crater targets (Fig. 4.23b) was scattered on the ground along the trajectory, arranged over four sub-fields. Altogether, 80 craters with diameters between 5 m and 0.2 m were used. The bigger craters were placed near the beginning of the path (higher altitudes) and the smaller craters nearer to the end (lower altitudes), ensuring a near-constant coverage of the camera images during the linearly decreasing altitude. After placing the crater targets, these were fixed to the ground by amassing soil along their circumference (Fig. 4.23b). A picture of the crater distribution is shown in Fig. 4.24.

Figure 4.24: Helicopter over crater field during flight test

4.3.1.2 Crater catalog

Subsequent to field preparation, a catalog of crater positions was created. The pose estimated by the Crater Navigation module is relative to this reference database. Tasks such as autonomous navigation for lunar landing or near-asteroid operation require the crater navigation to provide a pose in the reference frame of the target body. To reflect this principle, the crater catalog was expressed in the Earth-Centered Earth-Fixed (ECEF) reference system. A two-stage process was performed: At first, a tachymeter (Leica TDRA6000) was used to measure all crater centers and three auxiliary points in a local (tachymeter) reference frame. Then, using the Propak6, the same three auxiliary points were measured directly in ECEF. This allowed the determination of a transformation from the local tachymeter reference frame into ECEF. Applying this transformation to all measured craters yielded the ECEF crater catalog. The accuracy of this catalog is then at the level of 0.01 m to 0.02 m, limited by the imperfections of the manual process of pointing and steadying the laser reflector prism for measurement by the tachymeter, as well as by the measurement accuracy limit of the GPS/SAPOS system.

4.3.1.3 Ground truth

As mentioned above, a high-end GNSS receiver was used as means to obtain a *ground truth* for the tested trajectories. In an effort to increase the accuracy of this information, the output of the Propak6 receiver was fused with IMU data in post-processing. This not only smoothed the position and velocity solutions but also complemented the 2 DOF attitude information given by the receiver (pitch and heading) by the estimation of the roll angle. The slight observability of attitude provided by the accelerometer measurements in combination with measured position and velocity further

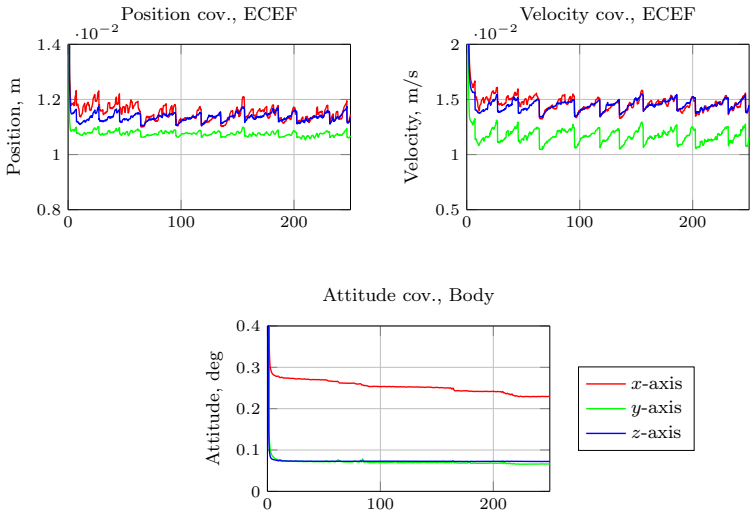


Figure 4.25: Fused ground truth quality (1σ covariance)

increased overall attitude accuracy. The covariance of the state dynamics for the fused ground truth can be seen in Fig. 4.25.

4.3.2 ATON Software Setup

For the flight experiments, several different software segments are involved. In the following the flight and ground segment of ATON are described.

4.3.2.1 Flight Software

The ATON software setup for the closed-loop flight experiments is shown in Fig. 3.55 on Page 83. Next to the ATON software modules, sensor drivers, a communication module to the helicopter and other supporting modules were added to the Tasking Framework. In detail, the flight software consists of the following tasks:

IMU Driver
The IMU driver's main responsibility is to receive and forward data from the iMAR iTraceRT-F400-Q-E IMU to the Navigation Filter task. For ATON this concerns the accelerometer and gyroscope raw measurements. Additionally, the internal calculated attitude solution of the iMAR iTraceRT is sent to a star tracker emulation. The Global Positioning System (GPS) measurements are not used by ATON software but are logged and used by the Result Evaluation task. The GPS time is used to synchronize the on-board clock with the GPS time.

Altimeter Driver (2x)
The altimeter drivers receive the measurements of the two laser scanners and convert their central beam measurements to an emulation of a single-beam altimeter, to be fused as a sensor within the navigation filter.

Camera Trigger Driver
A camera trigger task is responsible for triggering both cameras simultaneously with the selected image refresh rate. This task has a driver for the General Purpose Input/Output (GPIO) hardware of the visual flight computer that generates the trigger signals. This external trigger was chosen in order to achieve the highest possible accuracy in the assignment of time stamps to images. The time stamps of the transmitted triggers are provided to the camera driver tasks in order to add the trigger time to the corresponding image.

Camera Driver (2x)
The camera drivers are responsible for loading the images from the cameras and assigning the respective corresponding trigger time stamp and the system time stamp (time when the image is processed by the camera driver) to each image.

Star Tracker Emulator
Experiments by night with a star tracker mounted on the helicopter were not successful due to vibrations during flight. It was decided to emulate the star tracker by converting the attitude determined by the iMAR iTraceRT to the attitude with respect to ECI frame. The star tracker emulation task is executed for each time in flight where an STR measurement occurs. The calculation of the STR's attitude with respect to ECI is performed via a look-up table generated using NASA's SPICE Toolkit[1].

Undistort Filter (2x)
For each camera an undistort filter task is used to compensate for distortions caused by the lenses and the camera geometry. The camera calibration parameters were determined before the flight tests. The undistorted images are sent to the Crater Navigation and Binary Shadow Matching tasks.

Sample Rate Converter
A task is used to downsample the 400 Hz IMU messages to 100 Hz. The target frequency of the Navigation Filter is 100 Hz and the IMU message is used to trigger the Navigation Filter task. The IMU messages exhibit some jitter due to the Ethernet connection between IMU and visual flight computer. This jitter is compensated by triggering the downsampling task through a timed tasking event. When the task is executed, it accumulates all received IMU messages and forwards the downsampled value to the Navigation Filter task.

Feature Tracker (2x)
A feature tracker is available for each camera. For performance reasons, the feature trackers operate on the raw camera images (i.e. not the undistorted images). During the flight tests, only the feature tracker using the nadir camera images with a wider FOV was active.

[1] https://naif.jpl.nasa.gov/naif/toolkit.html

Crater Navigation (2x)
For each camera a Crater Navigation task is available that processes undistorted images. At each trigger time instant, the task also receives a navigation state estimate from the Navigation Filter to potentially speed up processing by predicting intermediate computation results and therefore enabling the Crater Navigation to skip those computations. To reduce the work load of the system, a parameter determines how often the Crater Navigation is triggered. For the flight tests the tasks were configured to run on an alternating schedule.

Binary Shadow Matching (2x)
Due to delays in the development of the flight version of the BSM, it was not used during the flight tests. Nevertheless, tests with data recorded during the previous open-loop flight experiments showed that the BSM is able to provide accurate pose estimation results with this kind of data. Here, the black parts of the crater patterns were used as shadows and the information from the crater catalogue was used to render the reference views.

Navigation Filter
The Navigation Filter task is triggered with 100 Hz with the downsampled IMU messages. Other messages, e.g. LA, Crater Navigation, etc., are considered as optional: they are not required for an update but taken into account if available at the beginning of any particulate Navigation Filter cycle.

Interface to Helicopter
The interface task to the helicopter sends the navigation solution of ATON to the flight computer of the helicopter and receives a status flag whether the helicopter uses its internal GPS solution or the ATON navigation solution for its autonomous actuator control.

Result Evaluation
The result evaluation tasks compares the ATON navigation state estimation with the GPS solution of the iMAR iTraceRT system. It also evaluates the covariances of the navigation filter. The task creates flags to indicate whether the ATON solution is within certain margins for a safe flight test. The flags are send to the ground station via the logger task. The following criteria have to be fulfilled for ten seconds before a flight test should be started:

- Standard deviation σ of the position estimation must be less than 5 m
- Standard deviation σ of the attitude must be less than 0.75 deg

During the flight the following error bounds are checked:

- Position error has to be less than 3 σ, horizontal error less than half of the altitude and the vertical error less than 5 % of the altitude or less than 0.5 m
- Velocity error has to be less than 3 σ
- Attitude error needs to be less than 3 σ and less than 0.4 deg

If any one of these conditions is violated then the closed-loop flight experiment is aborted by switching the navigation of the helicopter back to the GPS-based solution.

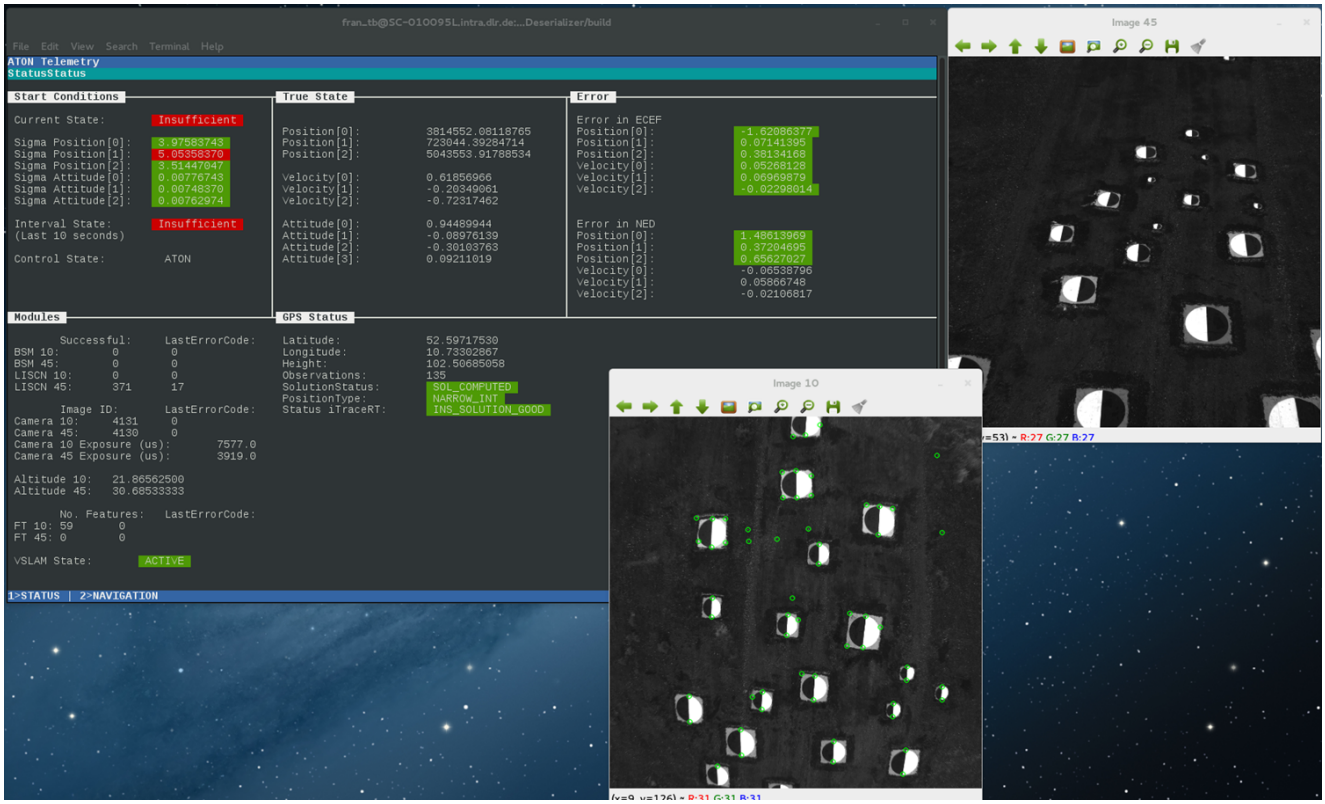


Figure 4.26: Screenshot of ATON's telemetry viewer at the ground station

Logger
The logger task collects all sensor data, intermediate results of certain modules for debugging purposes, the navigation filter output and the above mentioned flags. It stores all the data locally in text and image files. It also filters relevant data and sends it to the ground station via a UDP/IP link.

4.3.2.2 Ground Software

The ground software consist of two parts: commanding and telemetry viewer. The ATON flight software is telecommandeered by a direct connection to the visual flight computer on the helicopter based on a wireless Local Area Network (LAN). It allows the modification of some camera parameters like exposure time and activation of the time synchronization of the computer real-time clock with the GPS time. Additionally, the logging of the experiment is activated this way.

The telemetry viewer, as shown in the screenshot in Fig. 4.26, displays several measurements of the sensors, successful results of the image processing modules, camera images with detected features and the flags that are generated by the helicopter interface module and the on-board evaluation module. The flags are color-coded for easy assessment of the state of the sensors and the navigation state estimation of ATON during flight.

Parts of the telemetry viewer are also automatically generated by the MDSD framework, in particular the functions of deserializing the network telemetry data stream and of serializing the received telemetry for logging at the ground station.

4.3.3 Flight Test Results

During the flight test campaign described above, six single experimental flights were performed in closed-loop setup. For each flight the final altitude above ground was commanded individually. A safe final altitude of 0.75 m has been achieved, lower altitudes were avoided because of concerns about unknown behavior of the helicopter controller under the influence of strong ground effect. Figs. 4.27 and 4.28 show the track of the helicopter (ground truth and navigation solution) in the North-East and East-Up planes. The starting point of the trajectories is the point (0,0,0), where the helicopter hovers for a short time before initiating its descent. The vehicle then follows an almost straight path down to an altitude of about 10 m above the landing site. From that point, the helicopter executes a vertical descent down to the final altitude of about 0.75 m. In both plots it can be seen that the true trajectory (blue) and the navigation solution of the ATON system (green) differ only by a small amount.

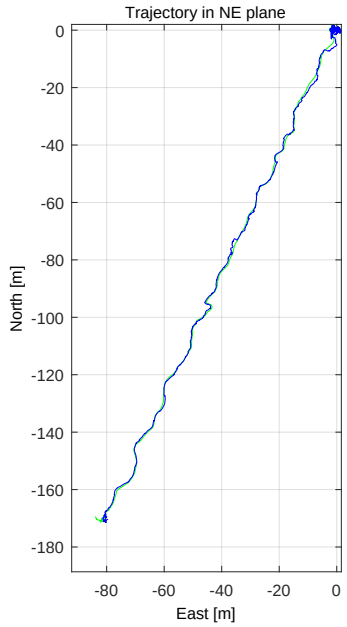


Figure 4.27: Plot of flight trajectory in North-East plane: blue - ground truth, green - ATON navigation solution; the experiment starts at the point (0,0)

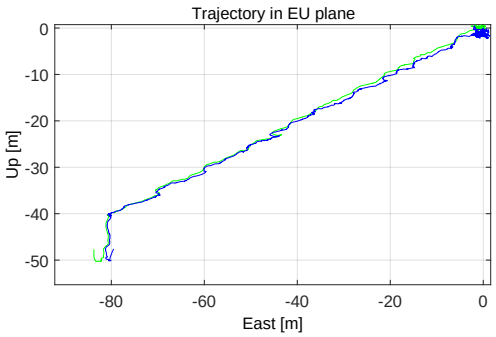


Figure 4.28: Plot of flight trajectory in East-Up plane: blue - ground truth, green - ATON navigation solution; the experiment starts at the point (0,0)

Figs. 4.29, 4.30, and 4.31 show the estimation errors for one of the closed-loop test flights. In Fig. 4.29 the estimated position errors and the estimated corresponding covariances are displayed. Furthermore, the rows of green, blue and yellow dots at –0.5, 0 and 0.5 m indicate the state of the navigation system and the state of the closed-loop guidance and control system. The blue and green dots at –0.5 m and 0.5 m denote an update of the navigation filter by the sensor inputs. The blue dots at 0.5 show updates by the forward camera. Green dots at –0.5 show updates of the down-looking camera. The yellow and green dots at 0 show which navigation solution is being used by the controller. Yellow dots indicate that the built-in GPS-based navigation system of the helicopter has been used. Green dots at 0 denote that the navigation solution of the ATON navigation system is used in closed-loop. The experiment stops at time 340 s. At that point the helicopter has reached the minimal safe altitude over ground. After that point the helicopter climbs up and guidance and control switches back to the GPS-based navigation solution.

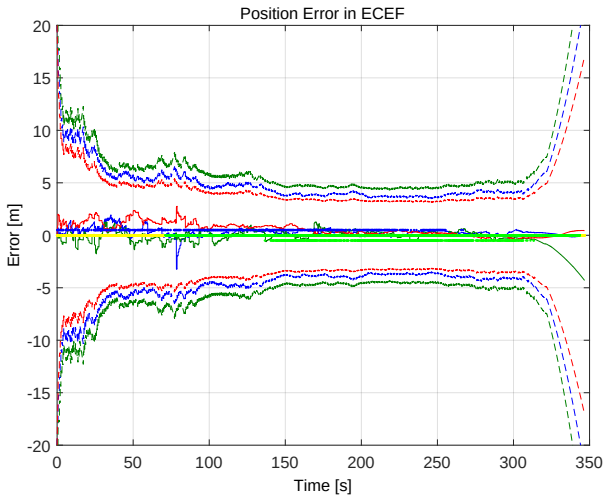


Figure 4.29: Position error in ECEF coordinates (x – blue, y – green, z – red); dashed lines denote the estimated error covariance; dots at -0.5, 0 and 0.5 denote the state of the system

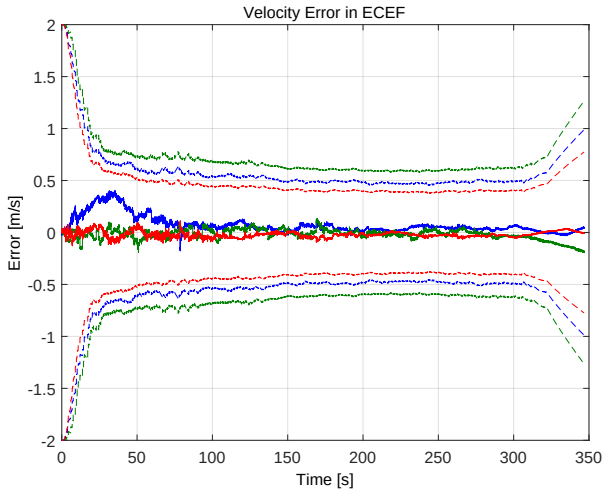


Figure 4.30: Velocity error in ECEF coordinates (x – blue, y – green, z – red); dashed lines denote the estimated error covariance

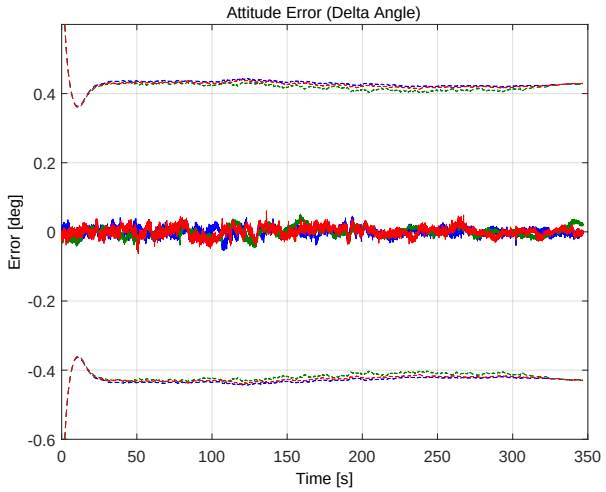


Figure 4.31: Attitude error in body-fixed frame (x – blue, y – green, z – red); dashed lines denote the estimated error covariance

When comparing the position estimation error and the covariances, it can be seen that at higher altitudes the position estimation is slightly worse than at lower altitudes. This is to be expected, since the patch of ground area resolved in any of the camera image's pixels is larger when at greater altitudes, meaning that the detected landmark positions are measured at lower effective resolution. Towards the end of the flight, when hovering low above the crater targets on ground, their visibility in both camera images is lost. At that point the absolute position updates to the navigation filter cease and the filter estimation error begins to grow again.

For the velocity errors in Fig. 4.30 a similar behavior can be observed. At high altitude, the errors are larger and shrink while the vehicle progresses towards lower altitudes. The error also starts to grow slightly when the observations from the image processing cannot be used for the filter update.

For the attitude error in Fig. 4.31 the deviations are independent from altitude as can be expected, since rotation-induced image displacement is invariant to the length of line of sight.

4.3.4 Conclusions for the Flight Tests

Progressing from HiL tests to flight tests, more variables and factors change than was the case when progressing from MiL to HiL tests. Additionally, the flight test setup introduces other and new limitations. Examples are the illumination conditions, sensor characteristics, as well as vehicle dynamics and motion that are qualitatively different from the selected reference scenario and from the previous conditions. While these limitations prevented covering all conditions of the reference scenario, the flight tests permitted covering and testing additional and previously untestable aspects.

One such new aspect was due to using and processing real measurements of all sensors, in real-time. This included errors in the intrinsic and extrinsic calibration of all sensors as well as digital communication and timing issues. Therefore, the development and demonstration of all capabilities needed to implement, calibrate and operate the whole ATON navigation system represented one of the most valuable outcomes of the flight tests.

The images and data generated during the flight test are extremely valuable for the further development of ATON. Development and maturation of the on-board software, including image processing modules and navigation modules, were advanced significantly by the need to use real measurements and images, and in particular by the need to process them in real-time. Effects such as errors and disturbances were present in the real data, eventually triggering results that cannot be reproduced in simulations. Therefore, we covered all technical aspects of a successful operation of the ATON navigation system. These include hardware handling and operation, software integration, real-time processing and operation, sensor calibration and alignment, ground truth measurement and processing, and preparation of the test range and of reference data.

As pointed out before, each test can cover only a part of the system under test, the expected environment and the operational envelope. For the flight tests in ATON, this means that the 3D chain could not be used in-flight, as the terrain lacks the significant altitude variation required by the method and no DEM data of the terrain is available. The other image processing elements could be used during the flight. The Shadow Matching was used in post-flight analyses. With the Navigation Filter module as the central element, we could prove that a real-time processing on board the test vehicle is possible. By tuning and enhancing all used modules, the navigation performance was improved during flights and by using PiL simulations in post-processing.

Finally, a high confidence in the robustness and performance of the ATON system was achieved. This allowed assessment of the risk of closing the loop and controlling the unmanned helicopter based on the navigation solution of the ATON system. In the final flight campaign, at first several open-loop flight tests were conducted to prove the navigation performance and robustness after enhancing the system with results from previous flights. Finally, the control loop was closed during the flight and the final altitude of the trajectory above ground was reduced step by step to less than 1 m. With these closed-loop flight tests it could be proven that the navigation performance, stability and robustness meets expectations and requirements. The results also show that closing the control loop does not affect the overall navigation system performance in-flight.

5 Lessons Learned

Within the project many lessons have been learned. The most important are summarized here:

- High-fidelity sensor simulation: For the proper development of image processing and navigation algorithms a thorough knowledge and a complete representation of sensor signals is needed. This includes the simulation of realistic images.
- Use and analyze real sensor data: For advancing the methods and algorithms as well as to make them more robust it is essential to switch to real sensor and image data at an early stage in development. This triggers failure modes which are not apparent in simulations. If this is done late the test with real data may contain a few surprises.
- Matlab/Simulink is very well suited to building complex simulation models and to easily integrate complex systems. However, integrating several C/C++ s-functions under development is not advised. If one s-function crashes, the whole Matlab process is aborted. The debugging of the s-functions is only possible if the whole Matlab process is executed in a debugger. For future projects, a decoupled development of C/C++ and Matlab/Simulink is advised, e.g., by communicating via a simple network protocol.
- Early real-time implementation: Implement from the start considering portability to embedded platforms (e.g. independent from libraries). If not considered the re-implementation for an embedded system comes at a high cost.
- Use model-driven software development: Since an optical navigation system, fusing optical and inertial sensor data, is a complex software, a model-driven software development approach is recommended [19]. This allows the interfaces of the single modules to be controlled and adapted in a consistent manner. In this way the tedious and time-consuming debugging of inter-module communication could be limited.
- Source code access: It is mandatory for the integration of such complex systems as ATON to have source code access to all code that has to be integrated. Solving integration problems on the basis of precompiled libraries is nearly impossible.
- Test in real-world and real-time environment: The transition to real sensors and real-time processing can lead to a number of pitfalls. If this could be done for parts of the system at an early stage it would reduce the effort for bug fixing when integrating and testing the complete complex system.
- In order to have more real-world test data available for upcoming projects, digital terrain models and crater catalogues should also be obtained for the powered descent model and the landing model in TRON.
- Accurate ground truth data: In order to assess the performance in HiL or flight tests, care should be taken to create a ground truth measurement with sufficient accuracy. It should be at least one order of magnitude better than the expected accuracy.
- Effort for test setup: Designing, implementing, and integrating test benches and flight test setups as well as executing the corresponding tests require considerable efforts and resources. When developing

new systems new means for testing and verification also have to be created. Then the development of a test bench becomes a complex project itself. This effort should not be underestimated

- Stereo matching can work for the reference mission but requires the lander's trajectory to be to be as parallel to the surface as possible. Analysis has shown that the stereo matching module can work in the altitude range of 10 km to 2 km with an average depth error of between 0.1 % to 0.5 % and an inter-image interval of approximately 10 seconds.
- The 3D Matching could be used from time to time as a final refinement step on pose estimates with a high confidence. It provides a high accuracy when given a good initialization, but due to the local optimization step in the ICP algorithm it is not well suited to correcting larger deviations of the actual pose.
- The Binary Shadow Matching can provide accurate pose estimation at mid and low altitudes. It works with deviations from the preplanned trajectory of 10 % of the line of sight distance while providing a pose estimation accuracy of 1% or less. The processing time is below 0.1 Hz and there is potential for further improvement. The BSM can cope with low resolution reference data with a resolution 12 times lower than the in-flight data. Hence it is usable shortly before the touch-down phase and thus a valuable alternative to the 3D processing chain.

6 Conclusions

This report provided an overview of the project ATON and its results from MiL tests via SiL and PiL tests to flight tests. The following sections summarize the main results and provide an outlook on current and future work and applications.

6.1 Main Results

With the last flight test campaign it was demonstrated that the ATON navigation system can provide a navigation solution for an exploration mission based on optical and inertial measurements in real-time. It could be proven that the provided navigation solution is accurate and robust enough to close the loop for the autonomous flight of an unmanned helicopter.

In addition to the system design and its verification the single modules of ATON reached a high maturity and TRL. The performances, robustness, and limitations for each specific module were tested and analyzed. Together with the results of the integrated ATON system this provides a thorough insight into the mechanisms and behavior of integrated optical navigation for exploration missions. This thorough knowledge as well as the corresponding source code is now available for the design and implementation of mission-specific optical navigation systems.

The ATON project also paved the way for verification of optical navigation sensors and components in representative environments. As such the creation of realistic scenes for cameras in the Testbed for Robotic Optical Navigation (TRON) and in flight tests on the unmanned helicopter have been major milestones and are now available for further development steps, for the verification of mission specific systems, and also for tests of equipment and software of the space community.

Throughout the project duration and while achieving several development milestones many valuable images, data, information, and lessons learned have been created, processed, and collected, respectively. They are also available for further development and verification activities.

6.2 Outlook

Although the project ATON has achieved a major milestone by demonstrating the capability of the navigation system to provide a robust and accurate navigation solution to guide and control an unmanned helicopter, the development of the system and its core software is continuing. Currently the focus is set on optimizing the software to make it more efficient and robust to run it on space-qualified hardware with limited computational resources. One element of this optimization is to transfer a part of the image processing to FPGAs. In parallel, the work is going on to adapt the system and its elements to different mission scenarios. They include asteroid orbiters and landers as well as landings on larger solar system bodies.

DLR is involved in several exploration missions, e.g. MASCOT with JAXA and the Interior Exploration using Seismic Investigations, Geodesy and Heat Transport (INSIGHT) mission with NASA. Since the exchange and partnering with European and international partner space agencies will continue for future exploration missions, several opportunities will arise to contribute the technologies of the project Autonomous Terrain-based Optical Navigation to joint missions.

Bibliography

[1] Nikolaus Ammann and Franz Andert. "Visual navigation for autonomous, precise and safe landing on celestial bodies using unscented Kalman filtering." In: *2017 IEEE Aerospace Conference*. Mar. 2017, pp. 1–12. DOI: 10.1109/AERO.2017.7943933.

[2] Farzin Amzajerdian, Diego Pierrottet, Glenn D. Hines, Larry Petway, Bruce Barnes, and John M. Carson. In: AIAA SPACE Forum. 0. American Institute of Aeronautics and Astronautics, Sept. 2016. Chap. Development of Navigation Doppler Lidar for Future Landing Missions. DOI: 10.2514/6.2016-5590. URL: https://doi.org/10.2514/6.2016-5590.

[3] Farzin Amzajerdian, Vincent E. Roback, Alexander Bulyshev, Paul F. Brewster, and Glenn D. Hines. In: AIAA SPACE Forum. 0. American Institute of Aeronautics and Astronautics, Sept. 2016. Chap. Imaging Flash Lidar for Autonomous Safe Landing and Spacecraft Proximity Operation. DOI: 10.2514/6.2016-5591. URL: https://doi.org/10.2514/6.2016-5591.

[4] Franz Andert, Nikolaus Ammann, Stefan Krause, Sven Lorenz, Dmitry Bratanov, and Luis Mejias. "Optical-Aided Aircraft Navigation using Decoupled Visual SLAM with Range Sensor Augmentation." In: *Journal of Intelligent & Robotic Systems* 88.2-4 (2017), pp. 547–565.

[5] Franz Andert, Nikolaus Ammann, and Martin Laubner. "Camera-Lidar Navigation for Large Unmanned Aircraft: Modular System Design and Safe Closed-Loop Testing Methodology." In: *CEAS EuroGNC Conference*. 2017.

[6] Franz Andert, Nikolaus Ammann, and Bolko Maass. "Lidar-Aided Camera Feature Tracking and Visual SLAM for Spacecraft Low-Orbit Navigation and Planetary Landing." In: *CEAS EuroGNC Conference*. 2015.

[7] C. Atkinson and T. Kühne. "Model-driven development: a metamodeling foundation." In: *IEEE Software* 20.5 (Sept. 2003), pp. 36–41. ISSN: 0740-7459. DOI: 10.1109/MS.2003.1231149.

[8] Yaakov Bar-Shalom, X.-Rong Li, and Thiagalingam Kirubarajan. *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc., 2001. ISBN: 047141655X. DOI: 10.1002/0471221279. URL: http://doi.wiley.com/10.1002/0471221279.

[9] Paul J. Besl and Neil D. McKay. "A Method for Registration of 3-D Shapes." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 14.2 (Feb. 1992), pp. 239–256. ISSN: 0162-8828. DOI: 10.1109/34.121791. URL: http://dx.doi.org/10.1109/34.121791.

[10] Jean Bézivin. "In search of a basic principle for model driven engineering." In: *Novatica Journal, Special Issue* 5.2 (2004), pp. 21–24.

[11] Alan W. Brown. "Model driven architecture: Principles and practice." In: *Software and Systems Modeling* 3.4 (Dec. 2004), pp. 314–327. ISSN: 1619-1374. DOI: 10.1007/s10270-004-0061-2. URL: https://doi.org/10.1007/s10270-004-0061-2.

[12] John M. Carson, Carl Seubert, Farzin Amzajerdian, Chuck Bergh, Ara Kourchians, Carolina Restrepo, Carlos Y. Villalpando, Travis O'Neal, Edward A. Robertson, Diego F. Pierrottet, Glenn D. Hines, and Reuben Garcia. In: AIAA SciTech Forum. 0. American Institute of Aeronautics and Astronautics, Jan. 2017. Chap. COBALT: Development of a Platform to Flight Test Lander GN&C Technologies on Suborbital Rockets. DOI: 10.2514/6.2017-1496. URL: https://doi.org/10.2514/6.2017-1496.

[13] Y. Chen and G. Medioni. "Object modeling by registration of multiple range images." In: *Proceedings. 1991 IEEE International Conference on Robotics and Automation*. Apr. 1991, 2724–2729 vol.3. DOI: 10.1109/ROBOT.1991.132043.

[14] Jody L. Davies and Scott A. Striepe. "Advances in POST2 End-to-End Descent and Landing Simulation for the ALHAT Project." In: AIAA-2008-6938. San Diego, CA, USA: American Institute of Aeronautics and Astronautics, 2008.

[15] Sven Efftinge and Sebastian Zarnekow. "Extending Java–Xtend: a New Language for Java Developers." In: *PragPub, The Pragmatic Bookshelf* 30 (Dec. 2011), pp. 5–11.

[16] CD Epp, TB Smith, and H. NASA. "Autonomous Precision Landing and Hazard Detection and Avoidance Technology (ALHAT)." In: *2007 IEEE Aerospace Conference*. 2007, pp. 1–7.

[17] Richard Fisackerly, Alain Pradier, Bruno Gardini, Berengere Houdou, Christian Philippe, Diego De Rosa, and James Carpenter. In: AIAA SPACE Forum. 0. American Institute of Aeronautics and Astronautics, Sept. 2011. Chap. The ESA Lunar Lander Mission. DOI: 10.2514/6.2011-7217. URL: https://doi.org/10.2514/6.2011-7217.

[18] M. Fowler. "Domain-Specific Languages." In: Addison-Wesley Professional, 2010. Chap. Generation Gap, pp. 571–573.

[19] T. Franz, D. Lüdtke, O. Maibaum, and A. Gerndt. "Model-based software engineering for an optical navigation system for spacecraft." In: *CEAS Space Journal* (Sept. 2017). ISSN: 1868-2510. DOI: 10.1007/s12567-017-0173-5. URL: https://doi.org/10.1007/s12567-017-0173-5.

[20] E. Gamma, R. Helm, J. Ralph, and J. Vlissides. "Design Patterns–Elements of Reusable Object-Oriented Software." In: 1st ed. Addison-Wesley Professional, 1994. Chap. Structural Patterns, pp. 196–208.

[21] D. Garcia-Castellanos and U. Lombardo. "Poles of inaccessibility: A calculation algorithm for the remotest places on earth." In: *Scottish Geographical Journal* 123.3 (2007), pp. 227–233.

[22] D.K. Geller and D.P. Christensen. "Linear Covariance Analysis for Powered Lunar Descent and Landing." In: *Journal of Spacecraft and Rockets* 46.6 (2009).

[23] Sébastien Gérard, Cédric Dumoulin, Patrick Tessier, and Bran Selic. "19 Papyrus: A UML2 Tool for Domain-Specific Language Modeling." In: *Model-Based Engineering of Embedded Real-Time Systems: International Dagstuhl Workshop, Dagstuhl Castle, Germany, November 4-9, 2007. Revised Selected Papers*. Ed. by Holger Giese, Gabor Karsai, Edward Lee, Bernhard Rumpe, and Bernhard Schätz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 361–368. ISBN: 978-3-642-16277-0. DOI: 10.1007/978-3-642-16277-0_19. URL: https://doi.org/10.1007/978-3-642-16277-0_19.

[24] D. Gonzales-Arjona and et.al. "Advances in the Hardware/Software co-design for the Absolute and Relative Vision Based Navigation systems for the Lunar Landing Scenario." In: *66th International Astronautical Congress*. International Astronautical Federation, Oct. 2015.

[25] *GRAIL mission homepage*. June 2010. URL: http://science.nasa.gov/missions/grail/.

[26] R. M. Haralick, H. Joo, C. Lee, X. Zhuang, V. G. Vaidya, and M. B. Kim. "Pose estimation from corresponding point data." In: *IEEE Transactions on Systems, Man and Cybernetics* 19.6 (1989), pp. 1426–1446.

[27] J. Haruyama, T. Matsunaga, M. Ohtake, T. Morota, C. Honda, Y. Yokota, M. Torii, and Y. Ogawa. "Global lunar-surface mapping experiment using the Lunar Imager/Spectrometer on SELENE." In: *Earth, Planets and Space* 60.4 (Apr. 2008), pp. 243–255. ISSN: 1880-5981. DOI: 10.1186/BF03352788. URL: https://doi.org/10.1186/BF03352788.

[28] Grant H. Heiken, David T. Vaniman, Bevan M. French, and Harrison H. Schmitt. *Lunar Sourcebook*. Cambridge University Press, 1991, p. 756. ISBN: 0521334446.

[29] Heiko Hirschmüller. "Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information." In: *CVPR 2005*. Vol. 2. IEEE, June 2005, pp. 807–814. URL: http://elib.dlr.de/22952/.

[30] Heiko Hirschmüller. "Stereo Processing by Semi-Global Matching and Mutual Information." In: *in IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.2 (Feb. 2008), pp. 328–341. URL: http://elib.dlr.de/55367/.

[31] Heiko Hirschmüller and Tilman Bucher. "Evaluation of Digital Surface Models by Semi-Global Matching." In: *DGPF 2010*. July 2010. URL: http://elib.dlr.de/66923/.

[32] T. Ho, V. Baturkin, C. Grimm, J. Grundmann, C. Hobbie, E. Ksenik, C. Lange, K. Sasaki, M. Schlotterer, N. Termtanasombat, M. Talapina, E. Wejmo, L. Witte, M. Wrasmann, G. Wübbels, J. Rößler, C. Ziach, R. Findlay, J. Biele, C. Krause, S. Ulamec, M. Lange, O. Mierheim, R. Lichtenheldt, M. Maier, J. Reill, H.-J. Sedlmayr, P.-W. Bousquet, A. Bellion, O. Bompis, C. Cenac-Morthe, M. Deleuze, S. Fredon, E. Jurado, E. Canalias, R. Jaumann, J. Bibring, K.H. Glassmeier, D. Hercik, M. Grott, L. Celotti, F. Cordero, J. Hendrikse, and T. Okada. "MASCOT - The Mobile Asteroid Surface Scout onboard the Hayabusa2 Mission." In: *Space Science Reviews* (2016). URL: http://elib.dlr.de/107136/.

[33] B. Houdou. *Next Lunar Lander, Phase A Mission Study, Mission Requirements Document*. Tech. rep. NEXT-LL-MRD-ESA(HME)-0001. ESA, Oct. 2008.

[34] HSO-IL. *PILOT Phase B+ Statement of Work*. Tech. rep. ESA-LEX-PIL-Bplus-SOW-0001. ESA, July 2015.

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

[36] John M Carson III, Edward A Robertson, Diego F Pierrottet, Vincent E Roback, Nikolas Trawny, Jennifer L Devolites, Jeremy J Hart, Jay N Estes, and Gregory S Gaddis. "Preparation and integration of ALHAT precision landing technology for Morpheus flight testing." In: (2014).

[37] Muzaffar Iqbal, Muhammad Uzair Khan, and Muhammad Sher. "System Analysis and Modeling Using SysML." In: *IT Convergence and Security 2012*. Ed. by Kuinam J. Kim and Kyung-Yong Chung. Dordrecht: Springer Netherlands, 2013, pp. 1211–1220. ISBN: 978-94-007-5860-5. DOI: 10.1007/978-94-007-5860-5_145. URL: https://doi.org/10.1007/978-94-007-5860-5_145.

[38] Andrew E Johnson, Yang Cheng, James Montgomery, Nikolas Trawny, Brent Tweddle, and Jason Zheng. "Real-time terrain relative navigation test results from a relevant environment for Mars landing." In: *Proc. AIAA GN&C Conference*. 2015, pp. 2015–0851.

[39] Andrew Johnson and Tonislav Ivanov. "Analysis and testing of a lidar-based approach to terrain relative navigation for precise lunar landing." In: *Proc. AIAA Guidance Navigation and Control Conference (AIAA-GNC 2011)*. 2011.

[40] S. J. Julier and J. K. Uhlmann. "Unscented filtering and nonlinear estimation." In: *Proceedings of the IEEE* 92.3 (Mar. 2004), pp. 401–422. ISSN: 0018-9219. DOI: 10.1109/JPROC.2003.823141.

[41] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte. "A new approach for filtering nonlinear systems." In: *American Control Conference*. Vol. 3. June 1995, pp. 1628–1632. DOI: 10.1109/ACC.1995.529783.

[42] Simon J. Julier and Jeffrey K. Uhlmann. "New extension of the Kalman filter to nonlinear systems." In: *SPIE Proceedings Vol. 3068: Signal Processing, Sensor Fusion, and Target Recognition VI*. 1997, pp. 182–193. DOI: 10.1117/12.280797. URL: http://dx.doi.org/10.1117/12.280797.

[43] J. N. Kapur, P. K. Sahoo, and A. K.C. Wong. "A New Method for Gray-Level Picture Thresholding Using the Entropy of the Histogram." In: *Computer Vision, Graphics and Image Processing* 29.3 (1985), pp. 273–285.

[44] Manabu Kato, Susumu Sasaki, and Yoshisada Takizawa. "The Kaguya Mission Overview." In: *Space Science Reviews* 154.1 (July 2010), pp. 3–19. ISSN: 1572-9672. DOI: 10.1007/s11214-010-9678-3. URL: http://dx.doi.org/10.1007/s11214-010-9678-3.

[45] H. Kaufmann, M. Lingenauber, T. Bodenmüller, and M. Suppa. "Shadow-based matching for precise and robust absolute self-localization during lunar landings." In: *2015 IEEE Aerospace Conference Proceedings*. Mar. 2015, pp. 1–13. DOI: 10.1109/AERO.2015.7119045.

[46] L. Kneip, D. Scaramuzza, and R. Siegwart. "A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation." In: *CVPR 2011*. June 2011, pp. 2969–2976. DOI: 10.1109/CVPR.2011.5995464.

[47] Hans Krüger and Stephan Theil. "TRON - Hardware-in-the-loop test facility for lunar descent and landing optical navigation." In: *18th IFAC Symposium on Automatic Control in Aerospace*. Sept. 2010, pp. 265–270. ISBN: 9783902661968.

[48] Hans Krüger, Stephan Theil, Marco Sagliano, and Stephan Hartkopf. "On-Ground Testing Optical Navigation Systems for Exploration Missions." In: *9th International ESA Conference on Guidance, Navigation and Control Systems*. June 2014, pp. 1–17.

[49] Vincent Lepetit, Francesc Moreno"=Noguer, and Pascal Fua. "EP*n*P: An accurate $\mathcal{O}(n)$ solution to the P*n*P problem." In: *International journal of computer vision* 81.2 (2009), pp. 155–166.

[50] S. Li, X. Jiang, and T. Tao. "Guidance Summary and Assessment of the Chang'e-3 Powered Descent and Landing." In: *Journal of Spacecraft and Rockets* 53 (Mar. 2016), pp. 258–277. DOI: 10.2514/1.A33208.

[51] Martin Lingenauber, Tim Bodenmüller, Jan Bartelsen, Bolko Maass, Hans Krüger, Carsten Paproth, Sebastian Kuß, and Michael Suppa. "Rapid Modeling of High Resolution Moon-Like Terrain Models for Testing of Optical Localization Methods." In: *12th ESA Symposium on Advanced Space Technologies in Robotics and Automation*. June 2013.

[52] Enrico Lockner, Thimo Oehlschlägel, Stephan Theil, Matthias Knauer, Jan Tietjen, and Christof Büskens. "Real-time capable trajectory synthesis via multivariate interpolation methods for a moon landing manoeuvre." In: *CEAS Space Journal* 6.2 (June 2014), pp. 107–118. URL: http://elib.dlr.de/97751/.

[53] Bruce D. Lucas and Takeo Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision." In: *International Joint Conference on Artificial Intelligence*. 1981, pp. 674–679.

[54] *Lunar Orbital Data Explorer*. http://ode.rsl.wustl.edu/moon/.

[55] Bolko Maass. "Robust approximation of image illumination direction in a segmentation-based crater detection algorithm for spacecraft navigation." In: *CEAS Space Journal* 8.4 (2016), pp. 303–314.

[56] Bolko Maass, Hans Krüger, and Stephan Theil. "An edge-free, scale-, pose-and illumination-invariant approach to crater detection for spacecraft navigation." In: *Image and Signal Processing and Analysis (ISPA), 2011 7th International Symposium on*. IEEE. 2011, pp. 603–608.

[57] Olaf Maibaum, Daniel Lüdtke, and Andreas Gerndt. "Tasking Framework: Parallelization of Computations in Onboard Control Systems." In: *ITG/GI Fachgruppentreffen Betriebssysteme*. http://www.betriebssysteme.org/Aktivitaeten/Treffen/2013-Berlin/Programm/. Nov. 2013. URL: http://elib.dlr.de/87505/.

[58] Marco Mammarella, Marcos Avilés Rodrigálvarez, Andrea Pizzichini, and Ana María Sánchez Montero. "Advances in Aerospace Guidance, Navigation and Control: Selected Papers of the 1st CEAS Specialist Conference on Guidance, Navigation and Control." In: ed. by Florian Holzapfel and Stephan Theil. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. Chap. Advanced Optical Terrain Absolute Navigation for Pinpoint Lunar Landing, pp. 419–430. URL: https://doi.org/10.1007/978-3-642-19817-5_32.

[59] R.K. Mehra and J. Peschon. "An innovations approach to fault detection and diagnosis in dynamic systems." In: *Automatica* 7.5 (1971), pp. 637–640. ISSN: 00051098. DOI: 10.1016/0005-1098(71)90028-8. URL: http://linkinghub.elsevier.com/retrieve/pii/0005109871900288.

[60] A. Miguel San Martin, David S. Bayard, and et.al. "A MINIMAL STATE AUGMENTATION ALGORITHM FOR VISION-BASED NAVIGATION WITHOUT USING MAPPED LANDMARKS." In: *10th International ESA Conference on Guidance, Navigation and Control Systems*. June 2017.

[61] Multiple. *Apollo 11 Mission Report*. Tech. rep. MSC 00-171. NASA, 1969.

[62] Multiple. *Apollo 12 Mission Report*. Tech. rep. MSC 01855. NASA, 1970.

[63] Multiple. *Apollo Experience Report - Mission Planning for Lunar Module Descent and Ascent*. Tech. rep. NASA TN D-6846. NASA, 1972.

[64] Multiple. *Surveyor Program Results*. Tech. rep. NASA SP-184. NASA, 1996.

[65] Thimo Oehlschlägel, Stephan Theil, Hans Krüger, M. Knauer, J. Tietjen, and C. Büskens. "Optimal Guidance and Control of Lunar Landers with Non-throttable Main Engine." In: ed. by F. Holzapfel and S. Theil. Vol. Selected Papers of the 1st CEAS Specialist Conference on Guidance, Navigation and Control. Advances in Aerospace Guidance, Navigation and Control. Springer Verlag, 2011, pp. 451–463. URL: http://elib.dlr.de/74978/.

[66] *OpenCV Open Source Computer vision library. url: http://docs.opencv.org/.*

[67] C. Padgett, K. Kreutz-Delgado, and S. Udomkesmalee. "Evaluation of Star Identification Techniques." In: *Journal of Guidance, Control, and Dynamics* 20.2 (1997), pp. 259–267.

[68] C. Paproth, E. Schlüßler, P. Scherbaum, and A. Börner. "SENSOR++: SIMULATION OF REMOTE SENSING SYSTEMS FROM VISIBLE TO THERMAL INFRARED." In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. XXXIX-B1. July 2012, pp. 257–260. DOI: 10.5194/isprsarchives-XXXIX-B1-257-2012. URL: http://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XXXIX-B1/257/2012/.

[69] Diego F. Pierrottet, Farzin Amzajerdian, and Bruce Barnes. *A long-distance laser altimeter for terrain relative navigation and spacecraft landing*. 2014. DOI: 10.1117/12.2050481. URL: http://dx.doi.org/10.1117/12.2050481.

[70] Alexandre Pollini and Hans Krüger. *FOSTERNAV: Flash Optical Sensor for Terrain Relative Navigation*. Tech. rep. European Commission's Directorate-General for Industry and Enterprise, 2012. URL: http://elib.dlr.de/81156/.

[71] J. J. Ribarich. "Surveyor spacecraft landing accuracy." In: *Journal of Spacecraft and Rockets* 5 (July 1968), pp. 768–773. DOI: 10.2514/3.29355.

[72] Vincent E. Roback, Farzin Amzajerdian, Alexander E. Bulyshev, Paul F. Brewster, and Bruce W. Barnes. *3D flash lidar performance in flight testing on the Morpheus autonomous, rocket-propelled lander to a lunar-like hazard field*. 2016. DOI: 10.1117/12.2223916. URL: http://dx.doi.org/10.1117/12.2223916.

[73] R. Roll and L. Witte. "ROSETTA Lander Philae – Touch-down Reconstruction." In: *Planetary and Space Science* 125 (2016), pp. 12–19. URL: http://elib.dlr.de/103769/.

[74] S. I. Roumeliotis, G. S. Sukhatme, and G. A. Bekey. "Smoother based 3D attitude estimation for mobile robot localization." In: *IEEE International Conference on Robotics and Automation*. May 1999, pp. 1979–1986. DOI: 10.1109/ROBOT.1999.770398.

[75] S. Rusinkiewicz and M. Levoy. "Efficient variants of the ICP algorithm." In: *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*. 2001, pp. 145–152. DOI: 10.1109/IM.2001.924423.

[76] Marco Sagliano, Hans Krüger, and Stephan Theil. "TRON Tool: REPRESENTING A MOON LANDING SCENARIO IN TRON." In: *GLEX 2012*. May 2012. URL: http://elib.dlr.de/75853/.

[77] Sebastian Scherer. "Low-Altitude Operation of Unmanned Rotorcraft." PhD thesis. Pittsburgh, PA: The Robotics Institute, Carnegie Mellon University, 2011.

[78] *SELENE Data Archive*. http://darts.isas.jaxa.jp/planet/pdap/selene/index.html.en.

[79] Navid Serrano and Homayoun Seraji. "Landing Site Selection using Fuzzy Rule-Based Reasoning." In: *ICRA*. IEEE, June 20, 2007, pp. 4899–4904. URL: http://dblp.uni-trier.de/db/conf/icra/icra2007.html#SerranoS07.

[80] Jianbo Shi and Carlo Tomasi. "Good Features to Track." In: *IEEE Conference on Computer Vision and Pattern Recognition*. 1994, pp. 593–600.

[81] Malcolm D Shuster. "Kalman filtering of spacecraft attitude and the QUEST model." In: *Journal of the Astronautical Sciences* 38 (1990), pp. 377–393.

[82] Richard Szeliski. *Computer Vision: Algorithms and Applications*. 1st ed. Texts in Computer Science. Springer London, Sept. 30, 2010. ISBN: 978-1-84882-935-0. URL: http://www.ebook.de/de/product/19111262/richard_szeliski_computer_vision.html.

[83] ESA-NEXT Team. *Next Lunar Lander with in-situ science and mobilty: Phase A Mission Study, Statement of Work*. Tech. rep. NEXT-LL-SOW-ESA(HME)-0001. ESA, Nov. 2007.

[84] Stephan Theil. *AT-RYNR-TN-009: Definition des ATON-Systems*. http://sites.portal.dlr.de/ry/ATON/Projektberichte/AP%205100/AT-RYNR-TN-009_1-0_Definition_des_ATON-Systems.pdf.

[85] Stephan Theil and Leonardo Bora. "Beacons for supporting lunar landing navigation." In: *CEAS Space Journal* (July 2016). URL: http://elib.dlr.de/105766/.

[86] Douglas L Theobald. "Rapid calculation of RMSDs using a quaternion"=based characteristic polynomial." In: *Acta Crystallographica Section A: Foundations of Crystallography* 61.4 (2005), pp. 478–480.

[87] Nikolas Trawny, Joel Benito, Brent E. Tweddle, Charles F. Bergh, Garen Khanoyan, Geoffrey Vaughan, Jason Zheng, Carlos Villalpando, Yang Cheng, Daniel P. Scharf, Charles Fisher, Phoebe Sulzen, James Montgomery, Andrew E. Johnson, MiMi Aung, Martin Regehr, Daniel Dueri, Behcet A. Acikmese, David Masten, Travis O'Neal, and Scott Nietfeld. In: AIAA SPACE Forum. 0. American Institute of Aeronautics and Astronautics, Aug. 2015. Chap. Flight testing of terrain-relative navigation and large-divert guidance on a VTVL rocket. DOI: 10.2514/6.2015-4418. URL: https://doi.org/10.2514/6.2015-4418.

[88] Guilherme Fragoso Trigo, Bolko Maass, Hans Krüger, and Stephan Theil. "Hybrid Optical Navigation by Crater Detection for Lunar Pin-point Landing: Trajectories from Helicopter Flight Tests." In: (2017).

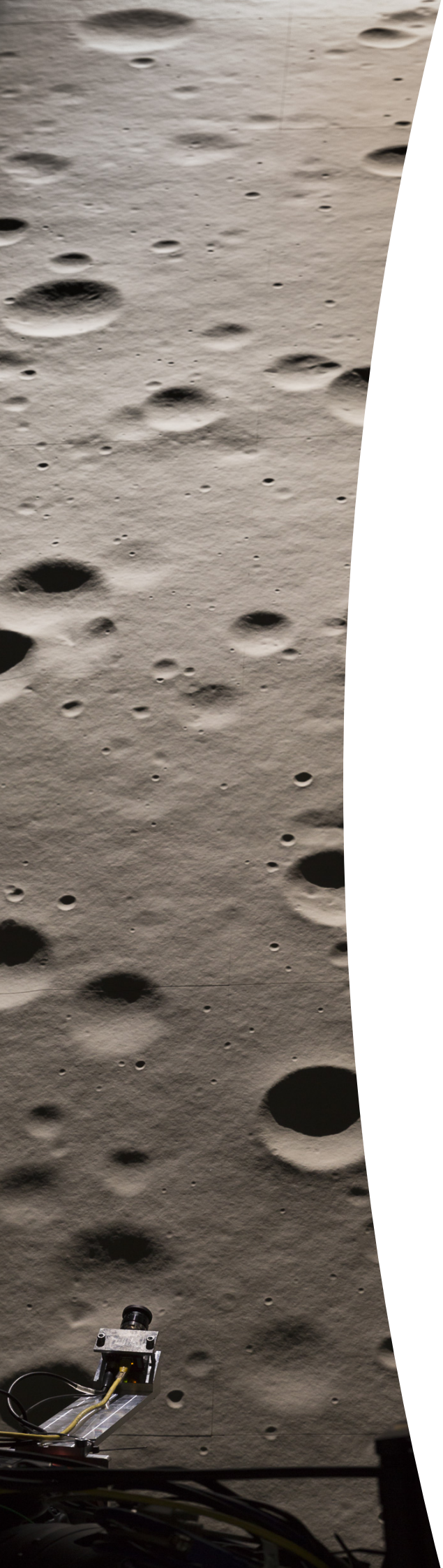
[89] Kevin Vipavetz, Thomas A Shull, Samantha Infeld, and Jim Price. "Interface Management for a NASA Flight Project using Model-Based Systems Engineering (MBSE)." In: *INCOSE International Symposium*. Vol. 26. 1. Wiley Online Library. 2016, pp. 1129–1144.

[90] E. A. Wan and R. Van Der Merwe. "The unscented Kalman filter for nonlinear estimation." In: *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium*. Oct. 2000, pp. 153–158. DOI: 10.1109/ASSPCC.2000.882463.

[91] L. Witte, R. Roll, J. Biele, S. Ulamec, and E. Jurado. "Rosetta Lander Philae – Landing Performance and Touchdown Safety Assessment." In: *Acta Astronautica* (2016). URL: http://elib.dlr.de/103770/.

[92] Yuanxin Wu, Dewen Hu, Meiping Wu, and Xiaoping Hu. "Unscented Kalman filtering for additive noise case: augmented versus nonaugmented." In: *IEEE Signal Processing Letters* 12.5 (May 2005), pp. 357–360. ISSN: 1070-9908. DOI: 10.1109/LSP.2005.845592.

[93] HongHua ZHANG, Jun LIANG, and XiangYu HUANG. "Autonomous hazard avoidance control for Chang'E-3 soft landing." In: *SCIENTIA SINICA Technologica* 44 (Mar. 2014), pp. 559–568. DOI: 10.1360/092014-51.



The DLR Institute of Space Systems at a Glance

The Institute of Space Systems in Bremen designs and analyzes future spacecraft and space missions (launchers, orbital and exploration systems, and satellites), and assesses them with regard to their technical performance and cost. It applies state-of-the-art methods of multi-disciplinary engineering in system design and analysis – for example, a computerized system for concurrent design.

In addition, the Institute of Space Systems cooperates with other DLR institutes and research institutions to develop, build, and operate its own spacecraft and missions. These are used to conduct scientific investigations and technology demonstrations involving, for example, small satellites and planetary landers. The Institute is a center of excellence for systems engineering with capabilities in system design, system integration, and systems testing, for which it plays a coordination and integration role.

At the Institute of Space Systems, research is also conducted into important system technologies, such as the behavior and handling of cryogenic fuels in tanks, landing technologies, attitude and orbit control systems, avionics systems, and high-precision optical measurement systems to enable future space missions or to improve existing technologies.

ISRN DLR-FB-2018-45

DLR **Deutsches Zentrum für Luft- und Raumfahrt**
German Aerospace Center

Institute of Space Systems
Robert-Hooke-Str. 7
28359 Bremen

DLR.de/irs