CEAS EuroGNC 2019
"Conference on Guidance, Navigation and Control"
3-5 April 2019 @ Politecnico di Milano, Milan, Italy.

CEAS-GNC-2019-049

Feature-based Optical Navigation for Lunar Landings

Magalhães Oliveira and Hans Krüger and Stephan Theil

1 Introduction

Future soft landing missions to celestial bodies will demand a high accuracy in absolute navigation. In such missions, the long signal delays, loss of accuracy with distance and the possibility of Line Of Sight (LOS) blocking don't allow for timely control decisions from the ground. Thus, to achieve high landing accuracy, the on-board Guidance Navigation and Control (GNC) system must be autonomous.

Absolute optical navigation uses optical information (such as camera images) of the celestial surface, combined with target specific information (such as a reference map), to determine the state of the Spacecraft (S/C) in a body fixed global frame (e.g., Moon Centred Moon Fixed). This paper describes an approach to absolute optical navigation based on image features. Although the method can be used in other celestial bodies, the remainder of the paper uses the lunar case as example.

1.1 Previous Approaches

Following is a brief overview of the methods used in past missions and of the current developments from National Aeronautics and Space Administration (NASA), European Space Agency (ESA) and Deutsches Zentrum für Luft- und Raumfahrt (Germany Aerospace Center) (DLR).

The first application of optical navigation for space exploration was done for the Mars Exploration Rovers (MER) in 2004. The Descent Image Motion Estimation System (DIMES) was developed by Jet Propulsion Laboratory (JPL) to address the challenges posed by the steady-state Martian winds [1]. By measuring the steady-state horizontal velocity, the Transverse Impulse Rocket System (TIRS) could be fired to compensate it. This system was, in fact, fired during the landing of the Spirit spacecraft allowing it to land safely.

Magalhães Oliveira
DLR, Address of Institute, e-mail: ana.magalhaesoliveira@dlr.de

Hans Krüger
DLR, Address of Institute e-mail: hans.krueger@dlr.de

Stephan Theil
DLT, Address of Institute e-mail: stephan.theil@dlr.de

There were three images used. In the first image, a high-contrast point was selected. The same point was found by correlation in the second image. The displacement between the points in the two images was used to determine the velocity of the lander. The same process was repeated between the second and third image. This method resulted in an estimate for the horizontal velocity of the lander with respect to the surface. However, by itself, it lacks the ability to determine an absolute position.

Currently, NASA is developing the Autonomous Landing and Hazard Avoidance Technology (ALHAT) project. Initially it was aimed at achieving safe precise landing on the Moon [4] and was later extended for any celestial body. It is to be implemented in the Mars 2020 lander [5] and for asteroids [12]. Two different approaches are included in this project.

The first approach is based on altimeter measurements. These are used to create a height profile of the surface in view of the S/C. The profile is matched to a Digital Elevation Map (DEM) of the surface [6], resulting in an estimate of the position in the body fixed frame (for the tests made, the Earth Centered Earth Fixed frame was used). This method requires an a priori knowledge of 1.6 km of the S/C's position. It was tested in 2012 and 2014 on the Vertical Testbed Morpheus.

The second approach is based on camera images. An initial guess is required, this time with an accuracy of 3 km in position and 0.15 deg in attitude [16]. The image taken by the camera is transformed to make it as if it was taken by a nadir-pointing camera (the image plane is parallel to the surface). The resulting image is correlated to a map of the surface to obtain the state estimate of the S/C. In 2014, this approach was tested on a helicopter flight and on Masten Xombie.

ESA is currently contributing to the Russian mission Luna-25 (aimed at the Moon) with Precise Intelligent Landing using On-Board Technology (PILOT) (a navigation concept) [3]. The method in development is based on camera images. The images are used to detect craters. A crater database prepared offline is projected onto the image, using an initial state estimate. This estimate must have an attitude precision of 0.1 deg [11]. The point cloud from the projection of the database is matched to the one from the detections. The match is then used to estimate the camera position and attitude.

Two implementations of PILOT are planned [3]. The first, to be launched on Luna-25, will serve as a hardware demonstration, whereas the second implementation, to be launched on a later mission, will be fully operational.

The Crater Navigation (CNav) algorithm is one of the technologies in the focus of the Autonomous Terrain-based Optical Navigation (ATON) program of the DLR [15]. In the scope of the CNav a sensor is developed which is capable of determining its own position and attitude with respect to any previously mapped cratered celestial body. The sensor works by detecting craters in an image it took of the underlying surface, and matching a constellation of these craters with an on-board database. Assuming the camera sees a part of the surface which is also mapped, this technology allows a real-time lost-in-space pose estimation of a spacecraft.

In a flight test with hardware-in-the-loop, CNav achieved a position accuracy of about 1-2 % of the LOS [17], defined as the length of the vector going from the camera origin along its optical axis to the intersection of this vector with the ground. Furthermore the CNav outputs were fused with Inertial Measurement Unit (IMU) measurements in an Extended Kalman filter algorithm. The result was used for the real-time guidance and control of the vehicle.

1.2 Paper Structure

A description of the approach in this study is provided in Section 2. Section 3 lays out the simulation environments used to obtain the results given in Section 4. These are discussed in Section 5 and used to derive the conclusions in Section 6.

2 Approach

Our approach to absolute optical navigation uses distinctive and georeferenced points or features on the surface of the target body (in this study, the lunar surface) to determine the absolute state of the camera (position and orientation in the Moon-Centred Moon-Fixed (MCMF) frame). An a-priori database of these points is obtained from processing images on ground, the navigation solution is obtained from matching features detected in images from the real flight with the ones stored in the database.

The database points are determined by feature detecting algorithms (feature detectors) applied on simulated images of the anticipated landing trajectory. The simulated images are rendered with a 3D computer model of the targeted surface, which is constructed with elevation information from a DEM. After their extraction, the descriptors of the features and their Cartesian position in the frame of the target body are stored in a database to be taken on-board the S/C.

The real camera on-board the S/C captures images during the landing. These images are provided as input to the software in real time. Features are found for each of these images. Through the descriptors, the features are matched to the most similar feature present in the database. Their coordinates are found in the database, resulting in pairs of 2D-3D coordinates (2D points from the real image, and 3D world points in $\mathscr{F}_{TRS}$ taken from the database), which are used to derive the camera pose (position and orientation) through the Effective Perspective n-Point (EPnP) algorithm [8].

Due to the mechanics of the descriptors there are requirements to the similarity of the images: the illumination conditions and the camera pose affect the appearance of the surface in the image, and consequently influence the set of detected features and also their descriptors. Therefore, the simulated images used to generate the database have to be to some extent similar to the images from the real flight. In a lunar landing, the illumination conditions can be determined a priori with the necessary accuracy. Although the trajectory that will be actually flown cannot be predicted, it is hypothesised that the expected perturbations from the nominal trajectory can be handled by the feature detectors. Thus, the nominal trajectory will be used to get the images used for the database.

In this paper we study the overall applicability of the approach. To do so we follow the following logic:

- define some reference trajectory (Subsection 3.4)
- generate database from images rendered using the nominal trajectory (Subsection 2.2)
- simulate real flight images using a terrain model in a laboratory environment (Subsection 3.1)

2.1 Feature Concept

For a computer, an image is a collection of pixels and their relative relevance is not obvious. Fig. 1 (left) shows three points in a simple image that provide different amounts of information, thus resulting in different grades of ambiguity. Point 1 contains no information; point 2 can be confused with any point on the square's sides; point 3 only has three other identical points in the image.

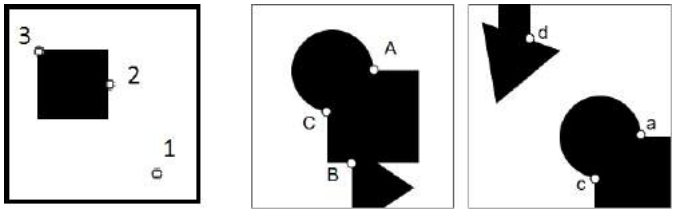

Fig. 1 Example of image points as features

The more unique and recognisable a point is, the easier it is to find it again in a new image and correctly match it. These points are referred to as image features. To be able to match features among images, each one is given a descriptor. A real world point found in two different images is expected to have very similar descriptors, which means the two features are likely to be matched across the images. For example, in Fig. 1 (middle and right), it is expected that the descriptor of feature 'A' is more similar to the one of feature 'a' than of any other feature in the image. Therefore, a match between 'A' and 'a' is likely.

There are various algorithms available that perform feature detection and description. From these, two were chosen based on their performance, speed and availability, namely Accelerated-KAZE (AKAZE) [2] and Binary Robust Invariant Scalable Keypoints (BRISK) [9]. The performance parameters considered were invariance to translation, rotation, scale, illumination changes and affine transformations, which were the transformations identified as relevant for the application. An algorithm is considered invariant to a transformation if it is successful in matching two images, where one is the transformed version of the other.

Differences between AKAZE and BRISK include the metrics used to detect a pixel as a feature, and the method behind the construction of the descriptor. Thus, using different algorithms will result in different points being detected. It was not known a priori if either feature detector was more suitable for images of the lunar surface. Since the programming commands needed are identical for both algorithms, including the option to switch between the two in the software did not add significant complexity, and allowed to search for a difference in the navigation accuracy between the two algorithms.

2.2 Database

To build the database the following data are used:

- 3D computer model of the target surface (the case studied was simply based on the DEM of the surface since its texture and optical properties did not significantly affect the images)
- Illumination conditions

- Poses of the camera during nominal trajectory
- Camera specifications (image resolution, focal length, camera intrinsic matrix, distortion parameters)

For each pose of the nominal trajectory, an image is rendered as if it was taken by the camera at that pose during the landing. The image renderer provides a visual image and the depth information associated. In practice, this means that, for each pixel of the visual image, we know the 3D coordinates of the point from the model that is represented by it.

A feature detector (AKAZE or BRISK in this study) is run on each image. For each detected feature, its 2D image coordinates and descriptor are obtained. Knowing which pixel corresponds to the feature, the 3D coordinates of the associated model point are taken from the depth information (as mentioned above).

The resulting database consists of the 3D coordinates and descriptors of features taken from every rendered image. The descriptors are organised (through the Fast Library for Approximate Nearest Neighbors (FLANN) algorithm [13]) in a structure that increases the matching speed.

2.3 Pose Determination

The approach is to determine the camera pose from the database. Every time an image is captured online, the same detector used for the database is applied on the captured image. For each feature found in the image, its 2D coordinates and its descriptor are obtained.

Then, the detected descriptors are matched to the database descriptors through the nearest neighbour (in terms of similarity of the descriptors) with FLANN. To minimise the number of incorrect matches, only the ones with the best similarity measure (smallest difference between the descriptors found in the image and the database) are used. From these matches, the 2D image coordinates (detected online) and 3D coordinates (database) are obtained. Then, by applying the EPnP algorithm, the came pose can be determined [8].

3 Test Setup

In this study several tests have been performed (described in Subsection 3.4).

First, the approach is validated using data corresponding to the best case scenario. For this test, the nominal trajectory will be used to generate the database and to capture the landing images. In other words, the database is generated from images that are as close as possible to the images captured by the camera during landing. In case this test fails, the approach can be safely considered unusable. The nominal trajectory was generated using the optimiser introduced in [14] and is illustrated in Subsection 3.4.

Second, it is also important to test a more realistic scenario. In a real landing, it is highly unlikely for the vehicle to follow exactly the nominal trajectory. To test these conditions, new trajectories were generated with the same optimiser used for the nominal trajectory (mentioned above), with different initial conditions (see Subsection 3.4). Since the test case was designed to represent a lunar landing, and for such missions the landing time (and as a consequence the illumination conditions) are precisely known, the effect of illumination perturbations were not inspected. This test was designed to determine whether the approach still results in accurate solutions if the input images are taken from poses deviating from the ones used for the database, in the range expected during a landing.

To perform the two tests described, the following requirements must be fulfilled:

- a terrain model representative of the lunar surface is available
- a DEM of the terrain model with sufficient resolution (see definition below) is available or can be generated
- it is possible to measure the position and orientation (pose) of the camera with higher accuracy than required by the navigation solution
- it is possible to either measure the camera pose directly in the model frame, $\mathscr{F}_{TRS}$, or to transform the measurements into this frame
- the lighting conditions can be easily reproduced in a rendering software

The DEM of the terrain model is considered of sufficient resolution if its effect is not visible in the rendered images. The lighting conditions shall be reproduced with a single source in the simulation and shall only require tuning of simple parameters such as the intensity.

3.1 TRON

Some of the requirements provided above were also considered when designing the Testbed for Robotic Optical Navigation (TRON) laboratory from DLR.

The simulation section of the TRON laboratory, shown in Fig. 2, includes a robot mounted on a rail running along the length of the room; a laser tracker system; three terrain models representative of the lunar surface; a control computer; and a lighting system.

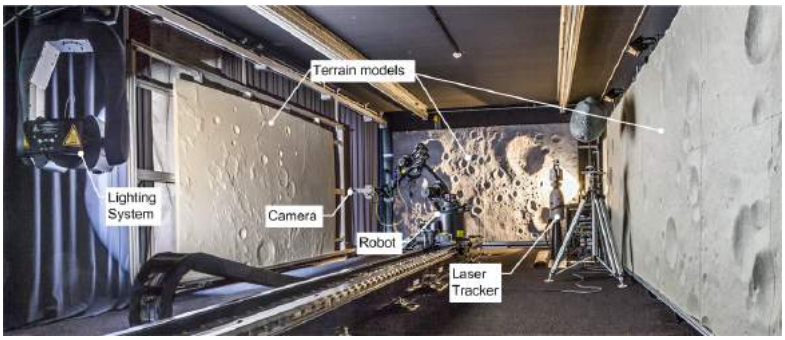

Fig. 2 Simulation section of TRON

From the available terrain models, the one facing the robot rail was chosen. It was the most suitable for the intended landing scenario. Furthermore, as the final set of its production, it received a manual finishing, which removed all visible milling lines and resulted in a practically infinite resolution. It is about 4.2×2.2 m and has terrain dynamics (difference between lowest and highest points of the model surface) of about 26 cm [10]. A DEM of this terrain model was provided. It had a resolution of 0.5 mm/pixel, which was found to be sufficient for the tests.

The KUKA KR 16 robot can be used to place the camera with six Degrees of Freedom (DoF) with respect to the robot's base. Additionally, the robot can move the base along the rail. The robot's hand holds the payload, which in this case is a plate where the camera and a 6 DoF target for the laser tracker (Tracker-Machine (T-MAC)) are mounted and fixed. The static

repeatability of the robot is 0.1 mm, which is compliant with the 2 mm accuracy required for the scaled navigation solutions [7].

The laser tracker (AT 901-MR from Leica) is an instrument used in the laboratory for precise measuring. It has a 6 DoF accuracy of about 15μm in position and 0.01° in attitude. For determining the ground truth for this test, it's accuracy complied with the 2 mm accuracy required for the scaled navigation solutions [7]. Reflectors can be used to measure position and the T-MAC to measure position and orientation.

The measurements are made in an internally defined frame, $\mathscr{F}_{LTS}$. Thus they have to be transformed into the model frame, $\mathscr{F}_{TRS}$. The $\mathscr{F}_{TRS}$ frame is defined by the position of three reflectors fixed to corners of the terrain model. By measuring their position with the laser tracker, the transformation from $\mathscr{F}_{LTS}$ to $\mathscr{F}_{TRS}$ is obtained. The DEM of the terrain has been obtained in the same reference ($\mathscr{F}_{TRS}$) by scanning it with the laser tracker system and a manually operated scanner. On the other side, the T-MAC and the camera are fixed on the payload plate. The pose of the camera is related to the T-MAC pose via hand-eye calibration. Thus, the setup allows to precisely measure the camera pose and transform it into the $\mathscr{F}_{TRS}$ frame.

In the laboratory, the use of curtains and black surfaces minimises the secondary light sources. The primary light source used was a simple construction site projector, the position of which was estimated by measuring a reflector. Although the simulation of the lighting conditions was not perfect, it was sufficient for what the test required.

3.2 Image Renderer

Blender was the software chosen to generate the 3D model from the DEM of the terrain model. The choice was based on its rendering quality, availability, intuitive graphical user interface and the possibility of automating the work-flow through python scripts. Apart from the 3D model, the automated rendering script uses a text file containing the camera poses to be used, namely the camera coordinates in the $\mathscr{F}_{TRS}$ frame and the quaternion of the camera frame as used in Blender.

Due to the simple optical properties of the material (e.g., colour and reflectivity), texturing the DEM with a grey matte surface was sufficent to render images which were very close in their appearance to the real ones.

3.3 Reference Frames

Figure 3 represents all the frames used in the laboratory and the sequence of transformation between them.

The terrain model frame, $\mathscr{F}_{TRS}$, is the frame in which the navigation solution and the ground-truth are given. It is defined by the three reflectors placed in the corners of the model. The origin is on the top-left reflector; the X_{TRS}-axis points along the vertical towards the bottom-left reflector; the Y_{TRS}-axis points to the right and is in the plane defined by the three reflectors and perpendicular to the X_{TRS}-axis; the Z_{TRS}-axis completes the right-hand rule. This is also the frame used to position the 3D model of the terrain, light source and camera in the rendering software (Blender).

The camera frame, $\mathscr{F}_C$, is used to define the camera pose in the laboratory. The navigation solution corresponds to the position and orientation of this frame with respect to the $\mathscr{F}_{TRS}$ frame.

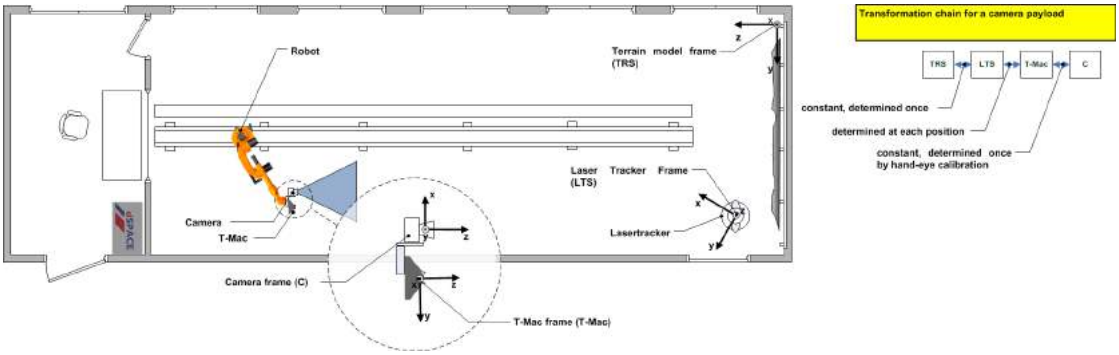

Fig. 3 Frames in the TRON laboratory and transformation chain between them

This frame is represented in Fig. 4 (left). The origin is at the camera's principal point; the Z_C-axis is along the principal axis pointing forward; the X_C-axis is along the horizontal side of the camera sensor pointing right; and the Y_C-axis completes the right-hand rule (along the vertical side of the sensor pointing down).

The rendering software used places the camera according to a different frame, $\mathscr{F}'_C$. This frame has the opposite direction of the Y- and Z-axis from frame $\mathscr{F}_C$, as shown in Fig. 4 (right).

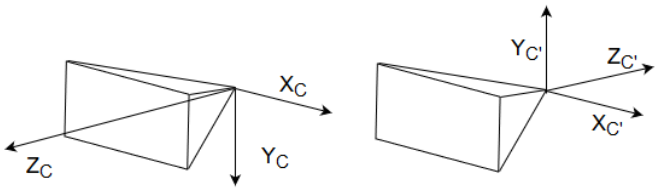

Fig. 4 Camera frames used to place the camera in the laboratory, $\mathscr{F}_C$ (left) and in the 3D software environment, $\mathscr{F}'_C$ (right)

The Laser Tracker (LT) frame is a proprietary frame defined by the manufacturer. The $\mathscr{F}_{LTS}$ frame is used only as an intermediate frame between the camera and model frames. Since the position of the LT in the laboratory can be changed to more optimally align it with the measurement ends (reflectors and/or T-MAC), the transformation from $\mathscr{F}_{LTS}$ to $\mathscr{F}_{TRS}$ needs to be determined every time it is moved by measuring the three reflectors of the terrain model.

Apart from the reflectors, the LT can also measure the T-MAC. This extension allows for 6 DoF measurements and was fixed with respect to the camera to measure its position and orientation. As for the $\mathscr{F}_{LTS}$ frame, the T-MAC frame, $\mathscr{F}_{T-Mac}$, is used as an intermediate frame for the transformation of the laboratory measurements. As with the LT the exact definition is proprietary. The transformation between it and the camera is obtained through hand-eye camera calibration and is fixed. The transformation to the $\mathscr{F}_{LTS}$ frame is given by each measurement made.

3.4 Datasets

Following is a description of the datasets used for the tests. For all datasets, a GC1380H camera was used. The camera resolution was 1360 x 1024 px, but the images were taken with a 1024 x 1024 px resolution. For datasets 0 and 1, a Cinegon 1.8/4.8 lens was used with about 67° of Field of View (FOV). For dataset 2, the lens used was a Cinegon 1.9/10 with FOV of around 35°.

Straight Line Trajectory – Dataset 0

Dataset 0 was used during development and for an initial validation of the software. The trajectory used is shown in Fig. 5 (left). It was a simple straight line trajectory with 61 points, starting at around 4 m and ending at around 1 m from the terrain model. The orientation of the camera remained constant, to simplify the relative positioning of the measuring instruments. The axis of $\mathcal{F}_C$ are represented for the first point of the trajectory using the RGB convention – red corresponds to the X_C-axis; green to the Y_C-axis; and blue to the Z_C-axis. The $\mathcal{F}_{TRS}$ frame is represented with black for the X_M-axis, magenta for the Y_M-axis, and cyan for the Z_M-axis.

Since the trajectory points were very close to each other and the orientation was constant, only 11 equally spaced points were used for the database to reduce its size.

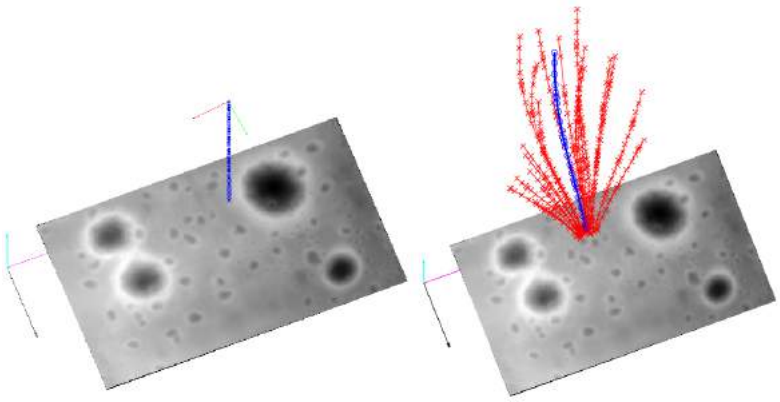

Fig. 5 Trajectories used during development, validation and tests; nominal trajectories in blue, perturbed trajectories in red.

Realistic Trajectories

To perform the proposed tests, a set of trajectories were generated using the same optimiser used for the ATON project [14]. The trajectories are represented in Fig. 5 (right). The nominal trajectory is represented in blue, with the points marked by circles. The perturbed trajectories are represented in red, with the points marked by crosses.

Nominal Trajectory

First, one reference (nominal) trajectory was determined. The simulated trajectory was designed to represent the final phase of the landing (powered descent) and to be compliant with the spacial restrictions of the laboratory (once scaled 1:100). In the local vertical frame, centred at the landing point, the nominal initial state was $\begin{pmatrix}-101 & 50 & 1500\end{pmatrix}^T$ m position and $\begin{pmatrix}10 & -10 & -40\end{pmatrix}^T$ m/s velocity. With this trajectory, a database was generated. This database was used for the validation and for the test with perturbed trajectories.

Perturbed Trajectories

The perturbed trajectories were created with the same optimiser by changing the initial conditions. The initial position was changed with random uniform error distributions of 100 m (for each coordinate) and the initial velocity with 10 m/s (for the magnitude of each component). These trajectories were used to determine whether the approach works for more realistic scenarios where the vehicle doesn't follow the reference trajectory as initially planned.

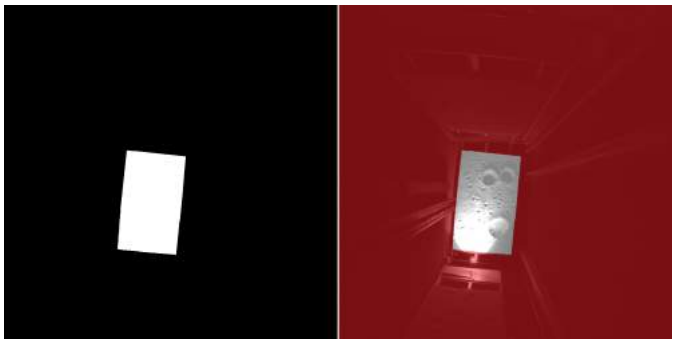

Fig. 6 Mask used to limit detections to the area of the terrain model

Different FOVs – Datasets 1 and 2

Once the trajectories were simulated in the laboratory, it was observed that, due to the limited dimensions of the model, the images captured from the farthest points had too little area corresponding to the model. Since in a real landing, most (if not all) of the area of the image is occupied by the surface, these images are not representative of a realistic scenario.

Masks were used to limit the search of features to the area of the image consisting of the terrain model. This way, no features are detected in the rails or other objects of the laboratory. To obtain the mask for each image, a new 3D model was used. This model consisted of a flat plane of the same size and position as the terrain model. The background colour was set to pure black and the intensity of the light source was set high enough for the plane to appear as pure white in the rendered image. An image was rendered for each camera pose and was later distorted to match the distortion observed with the real camera used (the distortion parameters were determined during camera calibration). An example of a mask is shown in Fig. 6 (left).

The example given is for the first image of the nominal trajectory. Fig. 6 (right) shows the mask overlaid on the image. The percentage of usable area of an image is defined as the percentage of white pixels of the total in the image. In the example, the percentage of usable area is about 5%.

This percentage is too low to be expected in a real landing. Therefore, only images for which a usable area was above 20% were used. The points used are represented in Fig. 7.

To be able to include points farther away from the model, where the deviation from the nominal trajectory is higher, a new set of images was taken with a different camera lens. The new lens had a smaller FOV, making the area occupied by the model in the image bigger. Hence, two datasets are defined: dataset 1 corresponds to the set generated with the large FOV camera; and dataset 2 was created with the small FOV camera.

Due to restricted laboratory time, only a limited number of camera positions could be used to get the real images for dataset 2. As such, the focus was on the points furthest from the model, which were not included for dataset 1. The trajectory points used, for both datasets, are shown in Fig. 7.

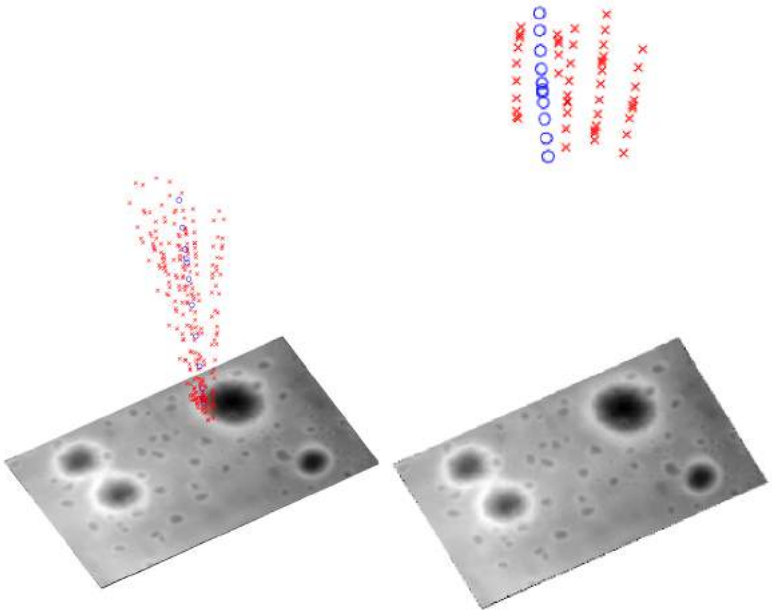

Fig. 7 Trajectory points used to capture the images for dataset 1 (left) and dataset 2 (right); nominal trajectories in blue, perturbed trajectories in red.

4 Results

The datasets described in Subsection 3.4 were used to validate the software developed (Subsections 4.1 and 4.2) and test whether the algorithm provides accurate solutions when the flown trajectory differs from the nominal trajectory in a range expected for a lunar landing (Subsection 4.3).

4.1 Straight Line Trajectory

The accuracy results obtained from dataset 0 are shown in Fig. 8 and 9 in the form of histograms, namely the position error norm (in grey); the position error in each world coordinate (difference between real vector and navigation solution); and the orientation errors of each axis. The position errors are given as a percentage of the LOS distance and are shown in the middle line. The orientation errors are defined as the angle between the real and calculated camera axis in question and are shown in the bottom line. The RGB convention was used.

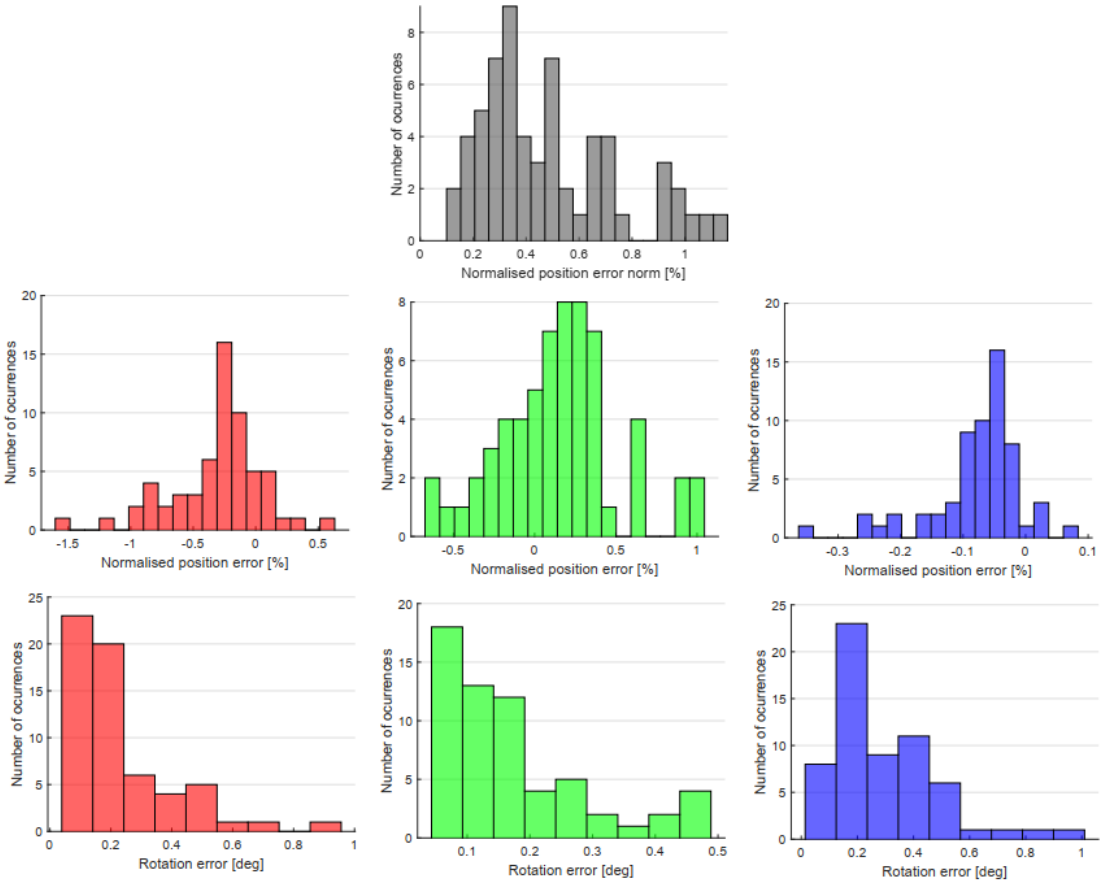

Fig. 8 Error histograms for navigation solutions using AKAZE on dataset 0, namely: norm of position error (grey); position error (second row) along the x (red), y (green) and z (green) axis; rotation error (third row) around the x (red), y (green) and z (green) axis

The navigation error for both AKAZE and BRISK had an average below 1% and a maximum error below 3%. The angle errors were lower for the y-axis and around the same magnitude for both the x- and z-axis. As for the position errors, it was lower in the z-direction and of the same magnitude for the remaining ones.

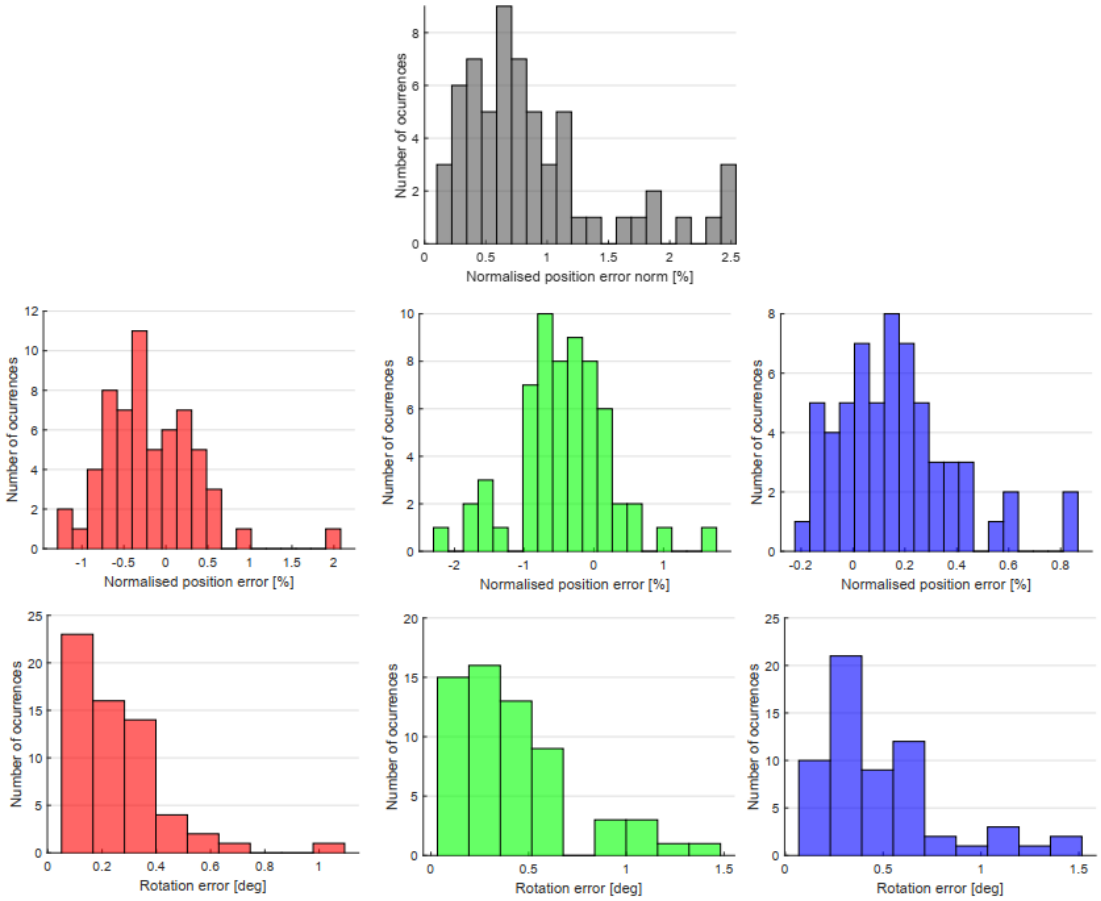

Fig. 9 Error histograms for navigation solutions using BRISK on dataset 0, namely: norm of position error (grey); position error (second row) along the x (red), y (green) and z (green) axis; rotation error (third row) around the x (red), y (green) and z (green) axis

4.2 Nominal Trajectory

The validation results using dataset 1 are represented in Fig. 10 and 11 for AKAZE and BRISK, respectively, with the same format used in Subsection 4.1. Fig. 12 and 13 show the same errors for dataset 2. The histograms for the angle errors were omitted, due to their correlations with the position errors(Subsection 5.1).

Once again, the magnitude of the z-direction error is lower than for the remaining two directions. The averages and maximum norms of the position errors are summarised in Table 1. It is important to note the small number of points available (12 for dataset 1 and 11 for dataset 2); as well as the existence of an outlier for the BRISK solutions on dataset 1.

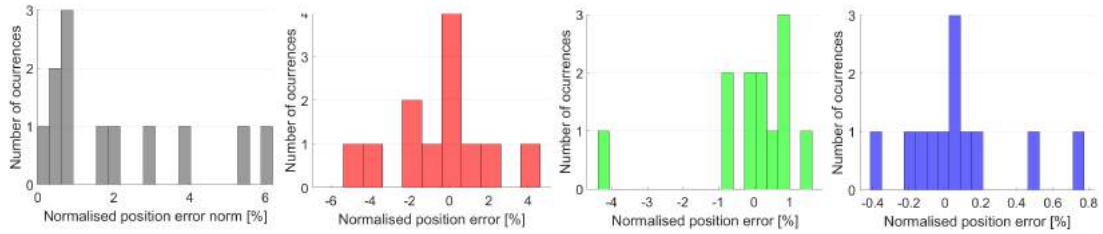

Fig. 10 Error histograms for navigation solutions using AKAZE on nominal trajectory of dataset 1, namely: norm of position error (grey); position error along the x (red), y (green) and z (green) axis

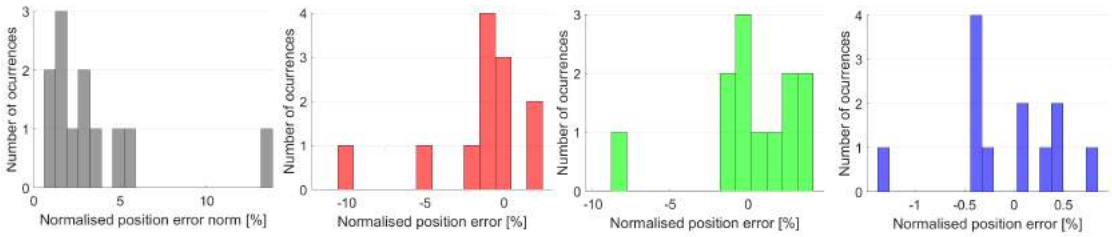

Fig. 11 Error histograms for navigation solutions using BRISK on nominal trajectory of dataset 1, namely: norm of position error (grey); position error along the x (red), y (green) and z (green) axis

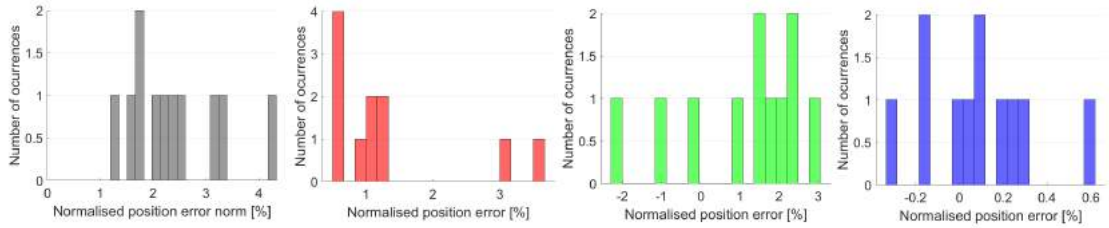

Fig. 12 Error histograms for navigation solutions using AKAZE on nominal trajectory of dataset 2, namely: norm of position error (grey); position error along the x (red), y (green) and z (green) axis

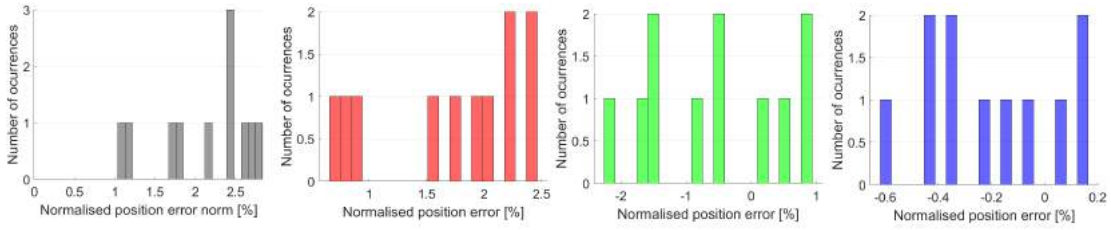

Fig. 13 Error histograms for navigation solutions using BRISK on nominal trajectory of dataset 2, namely: norm of position error (grey); position error along the x (red), y (green) and z (green) axis

	AKAZE		BRISK	
	Dataset 1	Dataset 2	Dataset 1	Dataset 2
Average	2.2%	2.4%	3.5%	2.1%
Maximum	6.1%	4.3%	13.8%	2.8%

Table 1 Average and maximum norm of position errors using nominal trajectory

4.3 Perturbed Trajectories

To check the performance of the algorithm for a realistic scenario, 20 trajectories were generated by perturbing the initial position provided to the optimiser and used to capture the online images. The database used was the same as the one used for Subsection 4.2. All trajectories were included in dataset 1, for a total of 239 points. The results are shown in Fig. 14 and 15 for AKAZE and BRISK, respectively. Due to limited laboratory time, only six of the perturbed trajectories were used for dataset 2, for a total of 50 points (Fig. 16 and 17). Table 2 summarises the averages and maximum norms of the position errors.

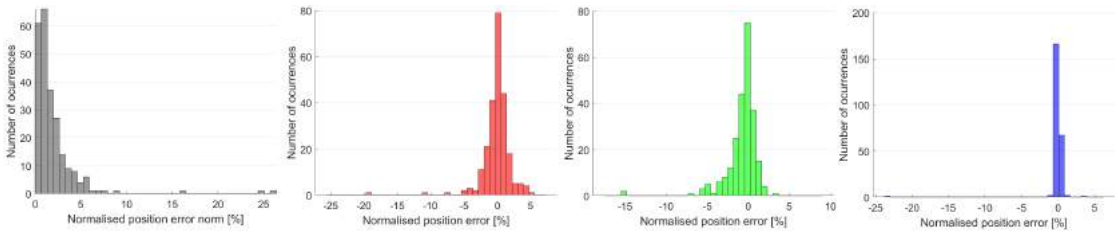

Fig. 14 Error histograms for navigation solutions using AKAZE on perturbed trajectories of dataset 1, namely: norm of position error (grey); position error along the x (red), y (green) and z (green) axis

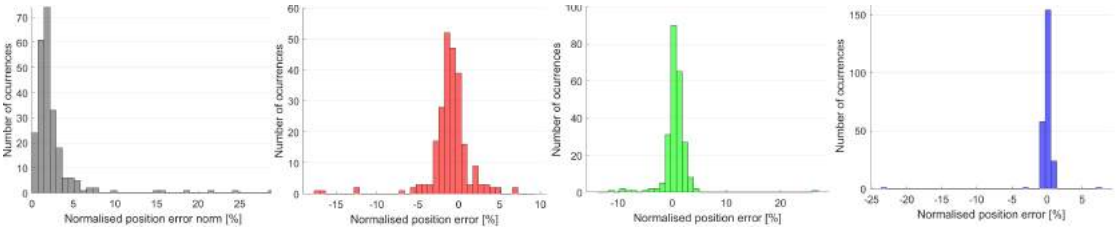

Fig. 15 Error histograms for navigation solutions using BRISK on perturbed trajectories of dataset 1, namely: norm of position error (grey); position error along the x (red), y (green) and z (green) axis

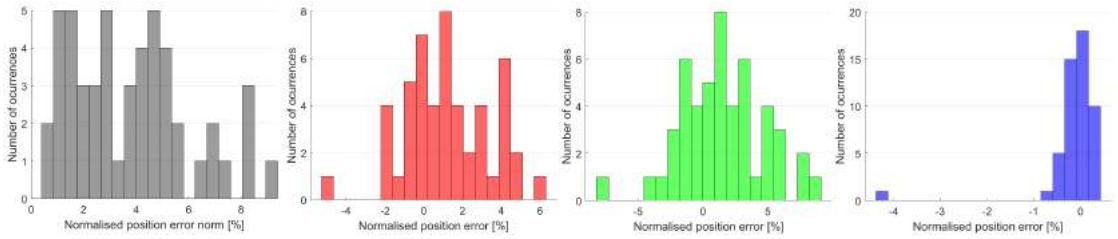

Fig. 16 Error histograms for navigation solutions using AKAZE on perturbed trajectories of dataset 2, namely: norm of position error (grey); position error along the x (red), y (green) and z (green) axis

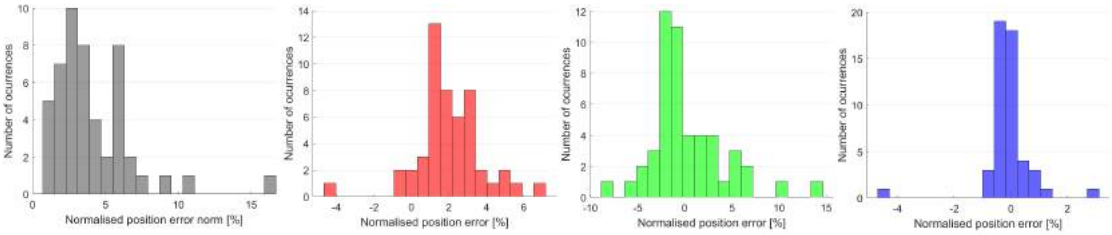

Fig. 17 Error histograms for navigation solutions using BRISK on perturbed trajectories of dataset 2, namely: norm of position error (grey); position error along the x (red), y (green) and z (green) axis

	AKAZE		BRISK	
	Dataset 1	Dataset 2	Dataset 1	Dataset 2
Average	2.0%	3.8%	2.5%	4.1%
Maximum	26.2%	9.2%	29.0%	16.6%

Table 2 Average and maximum norm of position errors using perturbed trajectories

5 Discussion

5.1 Position and Angle Error Correlation

EPnP calculates the position based on the solution obtained for the orientation, so the angle and position errors are correlated. This can be seen in Fig. 18 and was also observed in [17].

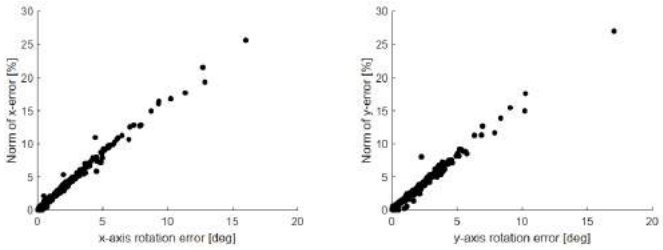

Fig. 18 Position error as function of orientation error

5.2 Position and Angle Errors

As mentioned in Subsection 4.1, the position error is lower for the z-direction of the camera frame than the remaining two directions. This held true for all the tests performed across both the datasets and detectors.

There was an increase of about 1% in the norm of the position error average between the straight line datasets and the nominal trajectory. For the nominal trajectory, the camera is positioned further from the terrain model than for dataset 0. With increasing distance, the individual pixel footprints grows, thus decreasing its resolution, which results in a larger position

error. This effect should be compensated when the error is normalised with the LOS distance. However, in the context of the laboratory, the terrain model is finite. Although the distance from the camera to the model (and, as a consequence, to the features) increases, the maximum possible distance between the features (in the real world) stays constant. In other words, as the camera gets further from the terrain model, its apparent size in the FOV gets smaller (see Fig. 6), and the geometry used to determine the camera's position gets worse, resulting in a higher error (relative to the LOS). The higher average error for the dataset 1 results with BRISK on the nominal trajectory was caused by one outlier: a 13% error in a set of 12 points contributes with more than 1% to the average (which is the difference observed).

We believe that the reason for the increase in the errors between the straight line datasets and the nominal trajectory lies in the geometry of the laboratory and therefore assume that with a bigger terrain model the errors would have been similar between them.

The results with dataset 1 showed no significant difference in the averages of the nominal and perturbed trajectories. The increase observed for dataset 2, suggests that it could be due to the higher difference between the flown trajectory and the nominal trajectory (as is visible in Fig. 5, the perturbations are larger at points further from the model). As we will show in the following, this is not the case. The position perturbation was quantified with two different measures and the error was plotted against them (see Figure 19) and against the angle difference.

First, each point in the perturbed trajectories was paired with a point in the nominal trajectory. Since both AKAZE and BRISK are less invariant to scale than to translation (meaning it is less likely for a feature to be correctly matched between images of different scales than between translated images), the paired point was the one for which the scale difference was smallest. In other words, the point for which the difference in Z_{TRS}-coordinates was smallest.

These paired point were then used to calculate the difference between them in each angle, the plane distance (difference between Z_{TRS}-coordinates) and the horizontal distance (distance between the $\begin{bmatrix} x_M & y_M \end{bmatrix}^T$ points). Both the plane distance and the horizontal distance were normalised with the LOS distance, to account for the fact that a 50 cm difference at 1 m from the model is much more severe than at 10 m from it. The absolute perturbation goes up to 0.3 m for the plane distance and 1.4 m for the horizontal distance. The angle differences were up to around 5 deg. Finally, the position error was plotted against these perturbation metrics for both datasets and detectors.

Fig. 19 shows the plots for the two distance measures. The angle plots were similar to the plot obtained for the horizontal distances. The plots show no clear relation between the perturbations in the trajectory and the error obtained. The existence of points with higher error for the lower values of plane distances is most likely a result of the much higher concentration of points in this range.

The stability of the error for the range of perturbations tested can be connected to the characteristics of the feature detectors used. Let camera $\mathscr{A}$ be the reference camera, positioned at the nominal pose. If camera $\mathscr{B}$ is positioned with the same orientation as $\mathscr{A}$, with the same Z_{TRS}-coordinate, but different X_{TRS}- and Y_{TRS}-coordinates (which relates to the horizontal distance), the difference between the images captured by each of the cameras will be translation. Therefore, the invariance of the feature detectors to translation results in the stability of the algorithm with respect to the horizontal distance.

If $\mathscr{B}$ is positioned at the same point as $\mathscr{A}$, but rotated around the Z_{TRS}-axis, the image captured by $\mathscr{B}$ will be rotated with respect to the one from $\mathscr{A}$. Once again, the detectors are invariant to rotation, so the error is stable for angle differences around the Z_{TRS}-axis. If the difference between the poses of $\mathscr{B}$ and $\mathscr{A}$ is only in the Z_{TRS}-coordinate, one image will be zoomed with respect to the other (different scales). The feature detectors are not completely scale invariant. However, there is invariance for small scale changes. Therefore, in this study, the

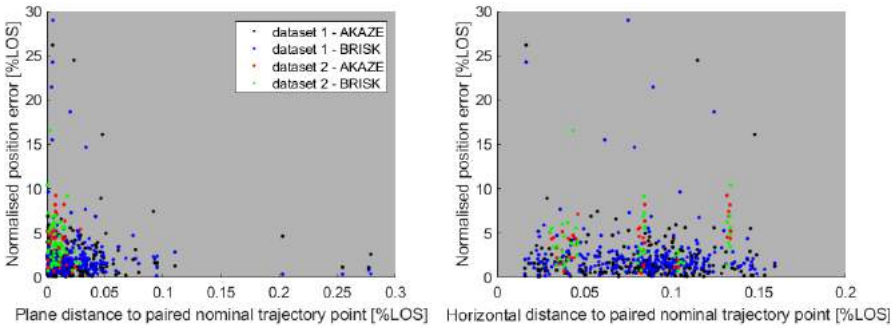

Fig. 19 Norm of position error as function of normalised plane distance (left) and horizontal distance (right)

database was built from images taken at different distances from the model. When $\mathscr{B}$ is rotated around the X_{TRS}- and Y_{TRS}-axis with respect to $\mathscr{A}$, the effect in the image is perspective change. The detectors are invariant to affine transformations. Although this does not equal invariance to perspective changes, for the range of angle differences observed in the tests (up to 5 deg) it is sufficient for stability.

6 Conclusions and Recommendations

The approach was successfully validated through the results presented in Subsections 4.1 and 4.2. These tests showed results consistent with the purpose of the algorithm: estimating the camera pose with respect to a terrain model representative of the lunar surface.

The results remained consistent with this purpose when the flown trajectory was perturbed with respect to the nominal trajectory from which the database was created (see Subsection 4.3). Thus all tests proposed were successful.

The performance of the algorithm (quantified by the norm of the position error as a percentage of the LOS distance) was not visibly affected by the perturbation of the trajectories. This is explained by the characteristics of the feature detectors, namely invariance to translation, rotation, and partially to scale and affine transformations. It also was not significantly different between AKAZE and BRISK.

The difference observed between the results from the straight trajectory (Subsection 4.1) and the nominal trajectory (Subsection 4.2) is a consequence of the spacial limitations of the laboratory. Due to the size of the terrain model, the geometry of the features that are used to determine the camera pose gets worse as the camera gets further from the model. This hypothesis should be tested with a larger terrain, for which the complete area of the image could be used for feature detection, using the same camera settings and distance to the model.

For future steps, the online software should be optimised and characterised in terms of computational load. The algorithms for selecting the features and matches to be used can be improved. The software should be tested in close loop. This could include a navigation filter with tight coupling of the features.

References

[1] Y. Cheng et al. “The Mars exploration rovers descent image motion estimation system”. In: IEEE Intelligent Systems 19.3 (May 2004), pp. 13–21.

[2] Pablo Fernández Alcantarilla. “Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces”. In: Sept. 2013.

[3] HSO-IL. PILOT Phase B+ Statement of Work. Tech. rep. ESA-LEX-PIL-Bplus-SOW-0001. ESA, July 2015.

[4] John M Carson III et al. “Preparation and integration of ALHAT precision landing technology for Morpheus flight testing”. In: (2014).

[5] Andrew E Johnson et al. “Real-time terrain relative navigation test results from a relevant environment for Mars landing”. In: Proc. AIAA GN&C Conference. 2015, pp. 2015–0851.

[6] 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.

[7] 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.

[8] Vincent Lepetit, Francesc Moreno”=Noguer, and Pascal Fua. “EPnP: An accurate $\mathcal{O}(n)$ solution to the PnP problem”. In: International journal of computer vision 81.2 (2009), pp. 155–166.

[9] S. Leutenegger, M. Chli, and R. Y. Siegwart. “BRISK: Binary Robust invariant scalable keypoints”. In: 2011 International Conference on Computer Vision. Nov. 2011, pp. 2548–2555.

[10] Martin Lingenauber et al. “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.

[11] Marco Mammarella et al. “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.

[12] 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.

[13] M. Muja and D. G. Lowe. “Scalable Nearest Neighbor Algorithms for High Dimensional Data”. In: IEEE Transactions on Pattern Analysis and Machine Intelligence 36.11 (Nov. 2014), pp. 2227–2240.

[14] Marco Sagliano et al. “On the Radau pseudospectral method: theoretical and implementation advances”. In: CEAS Space Journal 9.3 (Sept. 2017), pp. 313–331.

[15] Stephan Theil et al. “ATON (Autonomous Terrain-based Optical Navigation) for exploration missions: recent flight test results”. In: CEAS Space Journal (Mar. 2018).

[16] Nikolas Trawny et al. 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.

[17] Guilherme Fragoso Trigo et al. “Hybrid optical navigation by crater detection for lunar pin-point landing: trajectories from helicopter flight tests”. In: CEAS Space Journal (Jan. 2018).