Politecnico di Milano

Scuola di Ingegneria Industriale e dell'Informazione

Corso di Laurea Magistrale in Ingegneria Spaziale

Dipartimento di Scienze e Tecnologie Aerospaziali

Lunar Landing Navigation with Mono-camera

Relatore: Prof. Michèle LAVAGNA

Correlatore: Prof. Kazuya YOSHIDA

Tesi di laurea di:

Aureliano RIVOLTA Matr. 782471

Anno Accademico 2013–2014

To dad, grandparents
and all others
that aren't here anymore.

Acknowledgments

This work could not be completed without experimental activity held in the Space Robotics Laboratory of the Tohoku University, a Japanese university of Sendai city. This advanced laboratory is guided by professor Yoshida together with associate professor Nagatani and their assistant professors Sakamoto, Kuwahara and Nagaoka. A special thanks to Nathan J. Britton and Yudai Yuguchi for the very precious help in the development of the experimental set up, and all the students, starting from Daichi senpai, that worked in the lab every day easing the whole process.

The experience in Japan would not have been possible without the agreement resulting from the partnership of Prof. Lavagna and Prof. Yoshida, for that I am thankful.

This thesis would not have been readable and understandable without the very fast review of Paolo Lunghi, that has my most sincere thanks.

Fifteen man on the dead man's chest
Yo-ho-ho, and a bottle of rum!
Drink and the devil had done for the rest
Yo-ho-ho, and a bottle of rum!

Treasure Island, Robert Louis Stevenson

The (little) free time that I had back in Japan wouldn't have been the same without the people from the dormitory. I would like to thank Eri, Gabriele, Hosam, Jun, Masaru, Murat, Nevena, Sofia, and the others, whose name I probably never learned due to laziness.

Aside from the dormitory membership, I would like to thank John and Mingyo for the (multiple) gyoza-time, Wudom for the ramen, Fujii for the delicious tonkatsu, the Italians lost-in-Sendai Matteo, Monica, Martin and Teo, the Brazilian colony of Joao, Paula and their friends.

Friendships are to preserve forever, therefore special thanks to Ilaria, Marco, Martina, Marty, Melissa and Stefano for being there since years and years ago.

Per aspera ad astra

Hercules Furens, Lucio Anneo Seneca

This thesis on vehicle navigation wouldn't have been written by me if it wasn't for the precious experience in *Skyward Experimental Rocketry*. I would like to thank all the members of the past, present and future for the shared hopes and contagious passions. In particular my gratitude is for Giovanni and Michele for accepting me and giving me something I lacked: a dream to pursue. This little part of my life made me meet a lot of wonderful people like Ruben, Umberto, Mattia, Dario, Luca & Luca, Luigi & Luigi, Antonio, Gabriele, Francesco, Federico, Pietro... And for that I am grateful.

Many years later, as he faced the firing squad,
Colonel Aureliano Buendía was to remember
that distant afternoon when his father took him
to discover ice.

One hundred years of solitude,
Gabriel García Márquez

I apologize to my family for being absent during the development of this thesis. In particular I thank my mother Gabriella, my uncle Carlo and aunt Francesca and my great-aunt Agostina for bearing with my absence.

Abstract

Autonomous landing of space vehicles is a challenging problem and navigation is the most critical aspect. The application of computer vision in such problem is still a new and not fully tested approach, although being used for minor navigation tasks in other space missions.

The aim of this work was the development of a computer vision algorithm suitable for descent and landing phases of a moon lander using a single camera as main sensor. Generality, fast computation and reduced costs were the main driver for the navigation system design. The algorithm performance have been assessed in an experimental campaign carried out at the Space Robotics Lab in Tohoku University. Spacecraft dynamics and environment have been simulated by a camera and an inertial measurement unit, moved by a manipulator over a lunar surface reference.

The algorithm has been designed within the imposed limitations and the experimental activity increased the design robustness by showing a partial weakness in the computer vision algorithm that has been corrected accordingly. The experiments have shown that the proposed navigation algorithm is capable to reconstruct the motion of a lander with good precision even without the use of any filter, whose application could even increase such performance.

Sommario

L'allunaggio di veicoli autonomi è un problema stimolante il cui aspetto più critico è la navigazione. L'applicazione della computer vision è un approccio relativamente nuovo e non completamente testato, sebbene sia stato utilizzato per minori attività di navigazione in altre missioni spaziali.

Lo scopo di questo lavoro è stato lo sviluppo un algoritmo di computer vision adatto per fasi di discesa e allunaggio di un lander usando una camera singola come sensore principale. Generalità, computazione veloce e costi ridotti sono stati gli obiettivi primari per il design del sistema di navigazione. Le performance dell'algoritmo sono state determinate con una campagna sperimentale portata avanti allo Space Robotics Lab della Tohoku University. La dinamica del veicolo e l'ambiente sono stati simulati con una camera e un'unità inerziale mosse da un manipolatore verso una superficie lunare di riferimento.

L'algoritmo è stato progettato entro le limitazioni imposte e l'attività sperimentale ha incrementato la robustezza del design mostrando una parziale debolezza dell'algoritmo di computer vision che è stato concordemente corretto. Gli esperimenti mostrano che l'algoritmo di navigazione proposto è capace di ricostruire il moto di un lander con buona precisione anche senza l'uso di filtri addizionali, la cui applicazione potrebbe anche incrementare tali performance.

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

Roman Symbols

$\mathbf{a}_k$	LM state vector at step k
$A\left(u,\, v\right), B\left(u,\, v\right), C\left(u,\, v\right)$	Coefficients
$\mathbf{a}_i$	Inertial acceleration vector
$\mathbf{a}_m$	Accelerometer measure
$\mathbf{b}_a, \mathbf{b}_g$	Accelerometer and gyroscope biases
$c\left(u,\, v\right)$	Cross correlation
C	Consensus of a fitting
$d_{a,b}$	Distance from a to b
$d^2_{a,b}\left(u,\, v\right)$	Squared distance from a to b
d_k	Point k distance along a line from surface
$\mathbf{E}, E_{i,j}$	Essential matrix and its i,j component
$\mathbb{E}_{1,2}\left(u,\, v\right), \mathbb{F}_{1,2}\left(u,\, v\right)$	Autocorrelation between 1 and 2 (also rotated)
f	Focal length
$\mathbf{F}, F_{i,j}$	Fundamental matrix and its i,j component
$g\left(u,\, v,\, \sigma\right)$	Gaussian function
$\mathbf{g}$	Gravity force per unit mass
$\mathbf{H}$	Hessian matrix
$\mathbf{I}$	Identity matrix

$\mathbf{J}$	Jacobian
$\mathbf{K}$	calibration matrix
$\hat{\mathbf{l}}$	Unitary calibrated coordinate in terrain reference frame
$\mathbf{n}_a, \mathbf{n}_g$	Accelerometer and gyroscope measures noises
$\mathbf{P}, P_{u,v}$	Image and its intensity at coordinates u, $ypix$
$\mathbf{q}$	Quaternion vector
$\mathbf{Q}_1, \mathbf{Q}_2, \mathbf{Q}_3, \mathbf{Q}_4$	DCM partial derivatives w.r.t. quaternion
q_1, q_2, q_3, q_4	Quaternion component (q_4 scalar)
$\mathbf{r}$	Residual vector
$\mathbf{R}_{12}$	Rotation matrix (DCM) from 2 to 1
$\mathbf{R}_{tc,1}$	DCM from terrain to camera frame at instant 1
S	Matched points space
t_k, t_{k+1}	Time instants
$\mathbf{t}, \hat{\mathbf{t}}$	Translation vector (also unitary)
$\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}$	Single value decomposition matrices
u_0, v_0	Optical axis coordinates
u, v	Pixel coordinates
$\mathbf{V}_a$	Covariance matrix of $\mathbf{a}$
$\mathbf{v}_{l,1}$	Lander velocity vector at instant 1
$v_\downarrow, v_\uparrow$	LM damping parameter updater
$\mathbf{W}$	Weighting matrix
$\mathbf{w}_a, \mathbf{w}_g$	Accelerometer and gyroscope biases noise driver
$\mathbf{x}_{l,1}, Z_1, X_1, Y_1$	Lander position at instant 1 and its components
$x, y, \mathbf{d}$	Pixel directions and their vector
x, y, z	Rectangular coordinates
x_{tr}, y_{tr}, z_{tr}	Rectangular coordinates in terrain reference frame

$\hat{\mathbf{y}}$	Fitting function

Greek Symbols

$\chi^2\left(k\right)$	Chi squared criterion at step k
$\delta\mathbf{a}_k$	LM state vector increment at step k
$\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4$	LM Threshold
$\eta_{1,i}$	i-th calibrated camera coordinate vector of image 1
$\varphi\left(u,\, v\right)$	Image edge direction
$\gamma\left(u,\, v\right)$	Normalized cross correlation
$\kappa_k\left(u,\, v\right)$	Curvature index
λ_k	LM damping parameter at step k
ω_m	Gyroscope measures
$\mathbf{\Omega}_{pl}$	Planet angular velocity
ω_r	Relative angular velocity
$\rho\left(k\right)$	LM metric at step k
$\tau_{1,i}$	i-th feature depth of image 1
θ, x_e, y_e, z_e	Euler angle and Euler axis components
$\vartheta_x, \vartheta_y, \vartheta_z$	Rotation around axes x, y and z
$\xi_{1,i}$	i-th pixel coordinate vector of image 1
$\zeta_{1,i}$	i-th depth over translation magnitude of image 1

Acronyms

CF	Crater Field surface image
CFO	Crater Field Overexposed surface image
CV	Computer Vision
d.o.f.	Degrees of Freedom
EDL	Entry Descent and Landing
(E)KF	(Extended) Kalman Filter

SVD	Single Value Decomposition
GNC	Guidance Navigation and Control
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
LM	Levenberg-Marquardt (algorithm)
LP	Lunar Plains surface image
RANSAC	RAndom SAmple Consensus
SLAM	Simultaneous Localization And Mapping

Other Symbols

$[()_\times]$	Cross matrix
$[()_\times^\times]$	Double cross matrix
$[()_\otimes]$	Quaternion cross matrix

Chapter 1

Lunar landing optical navigation

1.1 Premise

Landing a vehicle on the Moon is not a trivial task, even if it has been done a lot in the past sixty years and more. The Moon represents still an important target for exploration, scientific research and colonization, therefore ensuring a safe and precise landing is mandatory. In case of human flight the lander can be not fully autonomous and be piloted by astronauts, like it happened for the Apollo 11 mission, however the general aim is to reduce the need of human intervention in such tasks by making the landing autonomous. Both manned and unmanned vehicles should not be piloted from Earth during the descent for numerous reasons: an automated Guidance Navigation and Control (GNC) system must be used.

The system guidance must determine the optimal trajectory and thrust profile in order to get to a safe and interesting landing point, but it needs a good navigation in order to do its job. Navigation is in fact the most critical aspect in the landing phase since the absence of precise information on ground distance and landing point could lead to a fatal failure.

The subject of this thesis work is the development and testing of a navigation algorithm for lunar landing[1] with a single camera as main sensor. Cameras offer a cheap system that can reduce weight, power and cost demand on the lander. The scope is thus to explore the possible application of mono-camera in the reconstruction of the lander state with the aim to have a robust, fast and cheap navigation system. Since a mono-camera cannot determine by itself a 6 d.o.f. motion model the use of auxiliary sensors is required to complete the reconstruction; anyway a limit in power, cost, weight and volume is implicitly intended.

The navigation algorithm need to be as general as possible, in order to be tested and

[1]In general the problem is under the Entry Descent and Landing (EDL) framework. In lunar landing only descent and landing phase are taken into account.

replicated on other systems. As a matter of fact such navigation can be theoretically used in other contexts, maybe removing some of the constraint imposed by the space mechanics (low computational power, very fast motion, kilometers of altitude, etc.).

1.1.1 System requirement

The navigation system that is investigated in this work must abide the following limitations:

- Use a mono-camera as the leading sensor,
- Use no map of the landing site,
- Track general features (craters excluded),
- Computational time as low as possible to be used on real space hardware.

These limitations define quite heavily the boundaries of this work and embody the principle of low-cost, general and easy to test navigation system. As shown in the following sections, these requirements obligate the algorithm to take a certain shape, leaving not much room for alternatives. The proposed algorithm have been tested with an experimental set up and a real camera in order to assess the robustness and performance achieved. Moreover the design of the algorithm should be intended to reduce as much as possible the computational time.

1.2 State of the art

1.2.1 Landing navigation systems

The first landing on an extraterrestrial body was performed in 1959 in the soviet Luna 2 mission with an impactor. The first soft landing was achieved in 1966 with the Luna 9 mission. At the time the most used sensors are radar altimeter and Inertial Measurement Units (IMUs), however the precision of the landing was really low. Landing on a pre-determined area was not easy due to IMU drift; higher precision was attained by the Apollo missions with the intervention of a human pilot. Navigation system performance were strongly limited by weak computational capabilities available in those years [1].

Radar altimeter were the main measure of distance to the ground and the guidance and control of the lander in such conditions relied more on this information than on IMUs measurements. The ranging measurement of radar altimeter are nowadays given to an evolution of laser altimeters (LIDARs), however they do not come at an easy price in terms of money, weight and power consumption. Lidars are studied a lot in lunar landing applications and seem a promising navigation tool that has less limitation than cameras, however the technological complication are still high and camera are a viable alternative for lunar landing. Nevertheless lidars can be coupled with optical devises to have a more robust navigation system.

1.2.2 Optical navigation in landings

In planetary landing camera measurement have been used by the Mars Exploration Rovers Spirit and Opportunity, although to reconstruct partially the lander state. In fact a downward camera has been used to determine the horizontal velocity on the lander since it was not possible to mount a Doppler radar by the time they found such information necessary. An algorithm called Descent Image Motion Estimation System (DIMES) [2,3] have been developed in order to check if a transverse impulse was needed in order to reduce horizontal velocity. The successful landing of Spirit, in which the system effectively entered in action, gave the basis for the use of camera in planetary landing [4].The DIMES was required to match images from highly different conditions in altitude and attitude. Despite the already proven technology, the DIMES approach is not suitable for a complete navigation.

In the HAYABUSA (MUSES-C) Japanese mission to asteroid Itokawa, an autonomous optical system for navigation and guidance has been used on board. Obtained results [5] have demonstrated that optical navigation is valid also in small body exploration. The technology readiness level of computer vision (CV) in landing scenario is not so high, however similar approaches have been carried out successfully, ensuring the appeal of such topic.

1.3 Literature review

The problem of the reconstruction of the state of a moving camera is in general called "ego motion" and has been studied for a long time. Camera can be used also to extract information about external environment: for example in planetary landing application images can be exploited to determine the hazard level of a landing site. Sometimes, if the feature tracked are craters or similar landmark, it is possible to fuse the hazard detection, hazard avoidance and navigation in one single algorithm. Since this was not the scope of this work, the hazard detection is left for further works.

The big limit of a single camera is that by itself it cannot sense depth, therefore all measures are affected by a scale factor. This implies that a single camera is not enough to produce a full navigation without additional inputs. One of this external input can be a map, a reference where landmarks such as craters are positioned.

In space exploration computer vision problems craters detectors have been studied intensively. In [6], for example, a Canny edge detector is used to generate an edge map, then clustering is applied to get preliminary points and crater radius. The final step makes use of a GVF-snake algorithm to detect the edges of the crater: this technique is often used for deformable shape recognition. Instead, in [7, 8], the algorithm searches for ellipses in the edge map and tries to refine and evaluate the results afterward. In [7] the navigation has been addressed using these landmarks as reference but not comparing them with a given map, instead a subset of craters and their relative position is exploited. In [9], after edge detection, a Hugh-transform based approach is enforced to detect ellipses on an asteroid.

The Vision Aided Inertial Navigation (VISNAV) [2] is a proposed navigation method that uses landmarks and reference map but also other sensors to increase robustness. Another effective procedure is fusion between vision algorithms and inertial navigation system to remedy the lack of information.

1.3.1 Optical-Inertial navigation

Inertial Navigation Systems (INS) have been used for a long time in navigation problems, therefore their properties and weaknesses are well known. INS can reconstruct the full 6 d.o.f. of a vehicle in an absolute (inertial) reference frame but due to error integration tends to suffer from long term drift. In order to determine the terrain relative motion, the angular velocity of the celestial body as well as its gravity field must be known beforehand, since accelerometers cannot measure gravity accelerations. Other sensors or an accurate motion model are required to reduce the drift. In lunar landing scenario, like for any Earth-based INS, the angular velocity and rotation motion is in general well known, of course this cannot apply for asteroid navigation. On Earth the most common complementary measure for the INS is the Global Positioning System (GPS) but in outer space this cannot be exploited, therefore camera aided INS has become a popular topic in the recent years [10–16]. One effective way to correct INS is to use an indirect Kalman Filter [17], thus modeling the INS error, and the rigid transformation between two camera views [15]. Most of these method augment the state with measured features, that is always computationally expensive. In ego motion about one hundred of tracked points are enough, therefore the already big state is augmented with twice or three times the number of feature found in the images [15]. As for other approaches, this kind of augmentation is not considered in this work since the computational time is a huge issue.

On the other hand, the use of the reconstructed 3-D motion between views seems an appealing choice, however one particular point must be stressed out: in order to

[2]for example [10], but the number of papers about VISNAV is higher and do not include more information for this work.

guarantee convergence, the camera reconstruction shall not be influenced by the INS. An error in the INS could cause an error in the computer vision algorithm resulting in divergence. In fact, some of the algorithms require an initial guess, that is better not to be based on INS if the intent is to correct the very INS reconstruction.

Another approach, exploited in [11], is to enforce a geometrical constraint known as the *epipolar constraint* (see Chapter 5), however this approach augment the system state, assume a perfect rotation reconstruction and has four numerical complications that may provoke the system failure. This constraint is deeply connected with the rigid motion, therefore the base is the same as the previous one.

Looking at the few comparable experimental activities presented in the aforementioned papers, it has been showed that coupling INS with camera measurement can reduce effectively the INS drift and be closer to the real path. In those papers where it is shown also the reconstruction by camera alone, it seems that the camera reconstruction is almost as precise as the filtered and coupled system, therefore a question about the need for inertial system coupling may arise. It is undoubtedly true that INS can provide an higher data rate than CV systems, however the INS gain more in terms of correction from the camera than the camera obtains from INS. In fact, the vision system without INS but with a sensor capable of getting depth, like an altimeter, can be used effectively as an appealing alternative to INS.

1.3.2 SLAM

Robot navigation has been studied for a long time but in recent years it has been found that good navigation can be achieved when also reconstructing a map of the environment. The problem of Simultaneous Localization And Mapping (SLAM) has been studied intensively in the last decades when it has been demonstrated its convergence [18, 19]. In the most naive implementation the SLAM based navigation algorithm uses a Kalman filter with a state augmented with the measured landmarks, suggesting an high computational cost. The idea is that the robot increase the precision of its positioning by successive measurement of landmarks, enabling the formation of a map or sub-maps [19] that can be re-used over time. One of the main branches in SLAM study is the EKF-SLAM where it uses an Extended Kalman Filter to handle non-linearity [20–23]. The other approach is called FAST-SLAM [19, 24] and make use of particle filters. A comparison can be found in [25] or related articles. Should be noticed that landmark acquisition include both optical and range measurement, therefore different kind of SLAM can be designed for the purpose.

In [21,22,26,27] just one camera and a SLAM approach have been used for navigation, in other cases inertial navigation is exploited [28], but the drawbacks of this approach make it not available option for lunar landing navigation. The computational cost is the main drawback of this kind of methods: while for surface rovers may be an appealing alternative, for the fast motion typical of a landing phase it is not. Moreover during landing it is unlikely for the lander to go back and re-visit a zone, therefore the map creation is useless for the majority of the travel. Should be noticed that this drawback is removed in case of small body navigation (asteroid, debris) or in cases where the lander makes numerous orbits before insertion in order to map possible landing sites.

1.3.3 Experimental validation

There are many papers about 6 d.o.f. reconstruction of a robot, however not many have been tested on real images and motion. Sometimes tests are conducted with limited motion and rotation and in almost all the cases the algorithms have been tested offline, therefore not involving real image capture, elaboration and state estimation on board. Many of the algorithms presented before have a non negligible computational burden that makes them not appealing for this work. In [15] experimental validation is carried out, but rotations are not included.

1.4 Thesis structure

This thesis starts with an introduction to reference frames, motion phases and inertial navigation system with Chapter 2. In Chapter 3 different camera models are presented and the strategy to get information from the sensors is described. A brief introduction to the image analysis problem is reported in Chapter 4 along with features and matching procedures. Chapter 5 is dedicated to the comparison of different algorithms suitable for the navigation system. The selected algorithm is studied in detail in Chapter 6 and a final navigation algorithm is proposed. The validation of such algorithm is handled in Chapter 7 through the results of experimental activity held in the Space Robotics Laboratory. Different motion sequences are analyzed, together with the sensitivity of certain parameter in order to address robustness and performance. A comparison with accredited papers is exploited to ensure the effectiveness of the algorithm. Conclusion and future improvements are presented in Chapter 8.

Chapter 2

Landing framework and INS

The first step towards the definition of a navigation system is the assessing of the framework in which the system is designed to operate. At the beginning a brief introduction to the landing phase of a space vehicle shows the motion range and the application boundaries of the camera. Then, different reference frames, needed to properly handle the problem, are introduced. The most used navigation system for landing, the INS, is presented at the end, putting emphasis on its errors and the studied remedies.

2.1 Landing phases

A landing maneuver can be divided in sub phases characterized by different motions and operative modes. For coherence the division described in [29, 30] is adopted:

- *Coasting phase*: in this phase the lander travel along a ballistic elliptical orbit. An attitude maneuver is performed to orient the main thrusters in the forward direction, in order to be able to reduce the velocity in subsequent phases. In many application the thruster positioning implies a complete 180° turn, therefore it is unlikely to use the camera for this particular maneuver. The altitude is lower than 5500 km.

- *Main brake phase*: the first part of the so called *powered descent phase* starts at the perilune of the elliptical orbit. Main thrusters are activated at the maximum available magnitude with the aim to drop the most part of the horizontal velocity. Navigation is of the utmost importance in this phase to assure that the correct profile given by the Guidance system is followed. At the end of this phase, at an altitude of about 2 km, the target landing site comes into the camera field of view.

- *Approach phase*: during this phase, the thrust is reduced in order to gain maneuverability. The lander motion is sufficiently smaller to be able to analyze the landing site for hazards and, if needed, plan one or two retargetings. Assessing the hazard level of the landing site is required in order to assure an autonomous and safe landing.
- *Terminal descent*: the last phase usually starts at very low altitude (tens of meters) and ends at touchdown, after a vertical descent at constant speed. Camera can still be used for navigation even if it is not strictly required.

From this basic phases division it is possible to address the range of use of the navigation system. Camera are mostly employed in the second and third phases where precise trajectory estimation and hazard avoidance must be performed. The altitude substantially changes during these phases, therefore the precision of the computer vision is expected to increase as that distance decreases. This is very intuitive and it is related to the size of the pixel: at high altitude it can represent a square of hundreds of meter, while in the final phase it could be less than one meter. This is strictly connected with the capabilities of the onboard camera.

2.2 Reference frames

In order to better understand the navigation problem a proper definition of reference frames is needed.

2.2.1 Moon-centric reference frame

This reference frame is centered on the center of mass of the body the lander is going to land, in this specific case the Moon. Axes are aligned towards fixed absolute directions (or their approximations over relative short time span, like the Gamma point). From the point of view of a lander, whose EDL phases last a very short time compared with the revolution period of the planet, this reference frame can be considered inertial and fixed.

2.2.2 Terrain relative frame

Since the Moon and in general celestial bodies are rotating while revolving around a main attractor, it is important to have a reference frame fixed with the body, rotating with it. This frame is of course not inertial, with all the complications that arises with that. Since there is freedom to define it, in this work this reference frame is considered to be fixed with the Moon surface and its origin is the vertical projection of the lander center of mass on the ground at the time instant the current landing phase begins.

2.2.3 Lander orbit frame

This reference frame is fixed with the trajectory of the lander, centered in its center of mass. The motion w.r.t. this frame is fast and not negligible, so it cannot be considered as an inertial frame. Usually axis are considered oriented toward the velocity vector and its orthogonal directions. This frame has not been considered in this work.

2.2.4 Lander principal axis frame

This reference frame is centered in the lander center of mass and aligned with its principal axis of inertia. It is particularly useful to describe the rotational behavior of the spacecraft.

2.2.5 Sensor reference frame

Should be noticed that some sensors measure quantities in their own reference frame and the rigid transformation between this reference frame and the principal axis frame must be known. In this work, the two reference frames (camera and principal axis) are considered as coincident, since the experimental activity aims to verify that the camera is able to understand its own motion. The generalization step should come afterward with little effort.

2.3 Inertial Navigation System

As stated back in Chapter 1, INS is one of the most exploited navigation technologies in lunar landing. For a better comprehension of the problematic of such systems, as well as possible remedy, a brief introduction is here reported. First of all an IMU consists of a gyroscope and an accelerometer, therefore it is important to understand what they measure.

2.3.1 Gyroscopes

Gyroscopes are sensors that measure the rotational velocity of the body they are attached to. Measures are given in the gyroscope reference frame but can be easily translated in the principal axis of the vehicle. These rotation velocity are absolute, meaning that in the case of the lander they detect rotation of the body w.r.t. an inertial frame like the one fixed with the Moon. These sensors are not capable by themselves to address the relative rotation of a body w.r.t. another rotating body. The measured angular velocity ω_m can be seen as a combination of planet rotation velocity $\mathbf{\Omega}_{pl}$ and the relative angular velocity ω_r, all expressed in the sensor reference frame.

$$\omega_m = \omega_r + \mathbf{R}_{si}\mathbf{\Omega}_{pl} = \mathbf{R}_{si}\omega_i \tag{2.1}$$

The planet rotational velocity vector is thus dependent on the relative orientation of the body thanks to the rotation matrix $\mathbf{R}_{si}$ that links an inertial reference frame with the sensor reference frame. ω_i is the inertial rotational velocity that is measured by gyroscope in their own reference frame fixed with the body. In order to get attitude information from these sensors are needed an inertial reference frame and the initial orientation of the vehicle in that frame. Then the measures of the gyroscopes are integrated with a proper procedure and attitude parametrization. By representing the lander attitude with a quaternion parametrization, the cinematic update equation is:

$$\dot{\mathbf{q}} = \frac{1}{2}\left[(\omega_m)_\otimes\right]\mathbf{q} \tag{2.2}$$

where the quaternion cross matrix for the 3-D vector ω_m is

$$\left[(\omega_m)_\otimes\right] = \begin{bmatrix} 0 & r & -q & p \\ -r & 0 & p & q \\ q & -p & 0 & r \\ -p & -q & -r & 0 \end{bmatrix} \tag{2.3}$$

with p, q and r that represent the scalar components of the body angular velocity vector ω_m in its own reference frame. This procedure can be carried out with gyroscope measures ω_m and a numerical integration methods for (2.2). Sometimes it is used also gyro measure interpolation, whenever this prove necessary.

2.3.2 Accelerometers

Accelerometers are sensors capable to measure the translational acceleration components of the vehicle they are mounted on, rotated in its own sensor reference frame. Moreover these sensors are not able to measure the gravity acceleration, therefore a free fall on the Moon would result in null measurement from them. Taking $\mathbf{R}_{is}$ the rotation matrix that rotates the sensor reference frame onto the chosen inertial reference frame and $\mathbf{g}$ as the gravity force per unit mass expressed in the same inertial reference frame leads to

$$\mathbf{a}_m = \mathbf{R}_{si}\left(\mathbf{a}_i - \mathbf{g}\right) \tag{2.4}$$

$$\mathbf{a}_i = \mathbf{R}_{is}\mathbf{a}_m + \mathbf{g} \tag{2.5}$$

Then the inertial frame acceleration $\mathbf{a}_i$ are integrated two times to get velocity and position. In this procedure an accurate gravity model is required, otherwise the integration would lead to huge errors. The problem is that such model depends on the distance from the center of the planet, i.e. the position, therefore an error in position would increase over time.

2.3.3 INS flow for inertial reconstruction

The procedure to reconstruct an inertial position makes use of the following formulation

$$\begin{cases} \frac{d}{dt}\mathbf{v}_{l,} & = \mathbf{R}_{is}\mathbf{a}_m + \mathbf{g} \\ \frac{d}{dt}\mathbf{x}_{l,} & = \mathbf{v}_{l,} \\ \frac{d}{dt}\mathbf{q} & = \frac{1}{2}\left[\left(\omega_m\right)_{\otimes}\right]\mathbf{q} \\ \mathbf{R}_{is} & = \mathbf{R}_{is}\left(\mathbf{q}\right) \end{cases} \tag{2.6}$$

with the following flow

Algorithm 2.1 INS - inertial

```
procedure INS-INERTIAL($\mathbf{a}_m, \omega_m, \mathbf{R}_{i0}, \mathbf{x}_{l,0}, \mathbf{v}_{l,0}$)

    loop
        $(\mathbf{a}_m, \omega_m, \mathbf{R}_{ik}, \mathbf{x}_{l,k}, \mathbf{v}_{l,k}) \leftarrow$
        $\mathbf{R}_{ik+1} \leftarrow (\omega_m, \mathbf{R}_{ik})$                    ▷ Update attitude
        $\mathbf{g}_k \leftarrow \mathbf{x}_{l,k}$                    ▷ Get the gravity model based on position
        $\mathbf{a}_{ik+1} = \mathbf{R}_{k+1i}\mathbf{a}_m + \mathbf{g}_k$                    ▷ Rotate accelerations
        $(\mathbf{x}_{l,k+1}, \mathbf{v}_{l,k+1}) \leftarrow (\mathbf{x}_{l,k}, \mathbf{v}_{l,k}, \mathbf{a}_{ik+1})$                    ▷ Integrate

        return $(\mathbf{R}_{ik+1}, \mathbf{x}_{l,k+1}, \mathbf{v}_{l,k+1})$
    end loop
end procedure
```

In order to reconstruct the relative state, it is possible to rotate afterward the inertial quantities into the rotating terrain frame or by incorporating the rotation velocity of the planet inside the whole process. The main source of error in INS is the drift of the state due to the integration of instrument noise and of course the most affected measure is the position since it relies on double integration and rotation.

2.3.4 INS for relative reconstruction - real model

In order to characterize the most used navigation tool in landing scenarios an error analysis is necessary. Taking into account the real measurements gives the following model

$$\begin{cases} \mathbf{a}_i &= \mathbf{R}_{is}\left(\mathbf{a}_m - \mathbf{b}_a - \mathbf{n}_a\right) + \mathbf{g} \\ \omega_r &= \omega_m - \mathbf{R}_{ai}\boldsymbol{\Omega}_{pl} - \mathbf{b}_g - \mathbf{n}_g \end{cases} \tag{2.7}$$

Where $\mathbf{n}_a$ and $\mathbf{n}_g$ are respectively accelerometer and gyroscope noise modeled as white noises (zero mean, Gaussian distribution) and $\mathbf{b}_a$ and $\mathbf{b}_g$ the biases modeled as integrated white noises with the simple model

$$\begin{cases} \frac{d}{dt}\mathbf{b}_a &= \mathbf{w}_a \\ \frac{d}{dt}\mathbf{b}_g &= \mathbf{w}_g \end{cases} \tag{2.8}$$

with $\mathbf{w}_a$ and $\mathbf{w}_g$ considered as white noises. In a very straightforward interpretation biases are low frequency high amplitude signals that corrupt velocity and attitude integration.

$$\begin{cases} \frac{d}{dt}\mathbf{v}_{l,} &= \mathbf{R}_{is}\left(\mathbf{a}_m - \mathbf{b}_a - \mathbf{n}_a\right) + \mathbf{g} - 2\left[(\Omega_{pl})_\times\right]\mathbf{v}_{l,} - \left[(\Omega_{pl})_\times^\times\right]\mathbf{x}_{l,} \\ \frac{d}{dt}\mathbf{x}_{l,} &= \mathbf{v}_{l,} \\ \frac{d}{dt}\mathbf{q} &= \frac{1}{2}\left[(\omega_m - \mathbf{R}_{si}\Omega_{pl})\times\right]\mathbf{q} \\ \frac{d}{dt}\mathbf{b}_a &= \mathbf{w}_a \\ \frac{d}{dt}\mathbf{b}_g &= \mathbf{w}_g \end{cases} \tag{2.9}$$

While (2.6) reconstructed the state in the inertial reference frame, (2.9)[1] reconstruct the relative state in the rotating terrain frame [31]. Should be noticed that here the angular velocity fed to the quaternion update is in the sensor frame, therefore relates the change of the latter with respect to the inertial and not vice versa. It is possible to reconstruct the relative attitude in two directions without problems.

A complete error model can be derived by subtracting the noiseless model to the real model. In the error model, since the quaternion parametrization has high mathematical complications, a small angle approximation is considered in attitude parametrization. Regardless of the error model it is important to notice the flow of the error.

[1]For better understanding of the matrices: $\left[(\omega_m)_\times\right] = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}$, $\left[(\omega_m)_\times^\times\right] = \begin{bmatrix} -q^2-r^2 & pq & pr \\ pq & -p^2-r^2 & qr \\ pr & qr & -p^2-q^2 \end{bmatrix}$

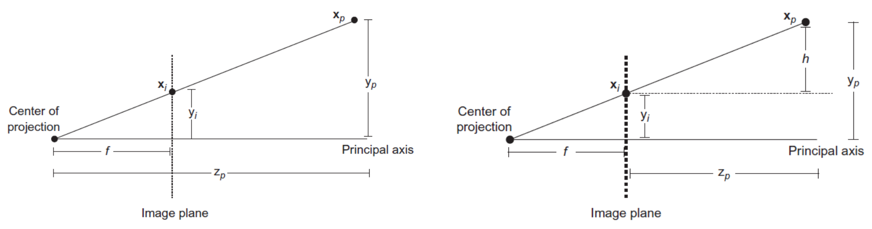
Biases increase the error on the long run and an error in attitude increase the error in velocity and position. The latter has the highest diverging rate since it integrates twice the accelerometer bias and attitude error caused by gyro bias.

The inverse Kalman filter approach takes the INS error model as the system to observe and requires the measurement of part of the error state. The error measurements are usually obtained by subtracting the INS integrator to the same measures obtained by different sensors. In fact the filter is coupled with an INS integrator that runs in parallel and takes the error determined by the filter as additional input for the following estimate. This strategy permit to use sensor with lower data-rate to correct the INS. It is important to analyze the observability of such error model in order to choose which kind of sensors to use: in general by measuring position and attitude it is possible to determine the state error and the biases [17]. In [15] a method to correct INS using computer vision reconstruction is proposed. If the rigid transformation between two camera views is available it is in general possible to address the whole correction, however this requires the computer vision algorithm to give information not based on INS, otherwise it could cause divergence. Recent studies by Roumeliotis et al. are focused on the observability analysis of such system [32–35].

Chapter 3

Camera models

In order to address the measurement coming from the camera, a proper introduction to the projection of a 3-D point onto an image is needed. For the sake of simplicity the camera models described in the textbook [36] are considered.



(a) Model 1 (b) Model 2

Figure 3.1: Projection of a point, from [36]

The principal axis depicted in both images of Figure 3.1 is often called *optical axis*. The image plane showed is the ideal plane collecting the light from the exterior, while in reality the Charged Coupled Device (CCD) collecting the image in its pixels is placed behind the center of projection and of course the image is rotated. The focal distance f is the distance between the point where all the projection meets, the center of projection, and the image plane (real or ideal is the same). Following Figure 3.1 notation, the projected point $\mathbf{p}_i = \begin{Bmatrix} x_i & y_i \end{Bmatrix}^T$ can be determined knowing the real point $\mathbf{p}_p = \begin{Bmatrix} x_p & y_p & z_p \end{Bmatrix}^T$. The two images represent two different parametrization of the same model. Using the projection of Figure 3.1a through triangular similarity:

$$\begin{cases} \frac{y_i}{f} = \frac{y_p}{z_p} \\ \frac{x_i}{f} = \frac{x_p}{z_p} \end{cases} \tag{3.1}$$

rearranging the terms

$$\begin{cases} y_i = f\frac{y_p}{z_p} \\ x_i = f\frac{x_p}{z_p} \end{cases} \tag{3.2}$$

Using the definition of Figure 3.1b leads to a similar form

$$\frac{y_i}{f} = \frac{y_p - y_i}{z_p} \tag{3.3}$$

$$y_i = \frac{y_p}{1 + \frac{1}{f} z_p} \tag{3.4}$$

therefore

$$\begin{cases} y_i = \frac{y_p}{1+\frac{1}{f}z_p} \\ x_i = \frac{x_p}{1+\frac{1}{f}z_p} \end{cases} \tag{3.5}$$

Both methods represent the projection, the only difference lies in the definition of z_p but since $f \ll z_p$ the difference between the two is very small. In case of lunar landing scenarios this is particularly true. Starting from these parametrizations three different camera models have been created: pinhole, affine and weak perspective.

3.1 Pinhole camera model

The pinhole camera model is one of the most used and relates the 2-D pixel measurement to a 3-D combination of parameters in the camera frame. Taking a point $\mathbf{p}$ in the 3-D reality and its respective equivalent in the unitary projective plane[1] and making use of (3.2)

$$\mathbf{p} = \begin{Bmatrix} x \\ y \\ z \end{Bmatrix} = z \begin{Bmatrix} x/z \\ y/z \\ 1 \end{Bmatrix} \tag{3.6}$$

However this still does not corresponds to the pixel coordinates $\begin{Bmatrix} u & v \end{Bmatrix}^T$ of the camera image. In fact x/z is a non dimensional ratio between two real world scale coordinates. The next step needs to take into account the camera optical properties like the focal length, the pixel dimensions and the optical axis. This can be done using the so-called intrinsic parameter of the camera gathered all in one transformation, giving birth to the intrinsic parameter or calibration matrix $\mathbf{K}$

[1]A vector in the projective plane is usually written as $\begin{Bmatrix} wx & wy & w \end{Bmatrix}$ with w an arbitrary number. Clarification and explanation can be found in a textbook like [36].

$$\begin{Bmatrix} x/z \\ y/z \\ 1 \end{Bmatrix} = \mathbf{K} \begin{Bmatrix} u \\ v \\ 1 \end{Bmatrix} \tag{3.7}$$

with

$$\mathbf{K} = \begin{bmatrix} \alpha_u & c & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

$$\alpha_u = f m_u \tag{3.9}$$

$$\alpha_v = f m_v \tag{3.10}$$

where $\begin{Bmatrix} u_0 & v_0 \end{Bmatrix}^T$ are the centroid (optical axis) coordinates[2], f the focal distance, m_u and m_v the coefficient that relate distance to pixels and c the skew coefficient between x and y axis of the image, often 0. The relation between a 3-D point $\mathbf{p}$ and its pixel coordinate projection ξ is

$$\mathbf{p} = z\mathbf{K} \begin{Bmatrix} u \\ v \\ 1 \end{Bmatrix} = z\mathbf{K}\xi \tag{3.11}$$

Therefore, if the camera calibration parameters and point depth are known it is possible to reconstruct the 3-D coordinates starting from the pixel coordinates on the image. Without the depth there is no way to get the 3-D coordinates but just their projective ratios.

3.2 Affine camera model

The affine camera model can be seen as a particular case of the perspective camera model where the focal length is taken as infinite. Taking (3.5) and setting the focal length to infinity gets to

$$\begin{cases} y_i = y_p \\ x_i = x_p \end{cases} \tag{3.12}$$

[2]In this form the coordinates are not expressed in pixels but in meter over pixel to be consistent

that simplify enormously the model by making it linear. However by doing so the depth information is completely lost and is no more possible to address translation in depth. Since this is a serious drawback in our scheme this model is not useful and therefore discarded.

3.3 Weak perspective camera model

This model can be seen as a scaling of the affine camera model and in general identified as a model between the first two. In this model equations (3.2) are approximated as

$$\begin{cases} y_i = \frac{f}{\overline{z_p}} y_p \\ x_i = \frac{f}{\overline{z_p}} x_p \end{cases} \tag{3.13}$$

where $\overline{z_p}$ is the average points depth. Basically the approximation lies in the consideration that all the points relative depths are negligible w.r.t. the focal length. This approximation can be considered valid when the camera is looking at almost planar surfaces, normal to the optical axis. In lunar landing the surface, especially from an high altitude, can be considered flat and the relative depth negligible: this requires a camera that is always nadir pointing. This possibility is not considered due to exaggerated constraint on the lander configuration, nevertheless it would simplify the discussion here after. The pinhole camera model is the only model valid for this work.

3.4 Camera calibration

The determination of the intrinsic parameters is necessary in order to have measures from a camera. This process is called *Calibration* and must be enforced whenever real imagery is involved. Even if a camera is mass produced it is always necessary to calibrate because small imperfections or variation can have a huge impact on the measures.

One of the nowadays most used techniques is found in [37]. The method is quite simple and flexible and has been used in many calibration tools software. This method requires to have different images from different viewpoint of a chessboard of known dimensions, the more the better. Chessboards are nice real-world object useful for calibration since a corner detector for each box is easily implemented and with good precision. In all the images is adopted a common reference frame fixed on the chessboard and used to get the real world coordinates by knowing the dimensions of each square. Then the 3-D rigid transformation from real coordinates to pixel coordinates is simplified using the reference frame of the board and an homography is computed using all the points detected in each image. Homographies are rigid transformation in homogeneous coordinates. Afterward the calibration matrix components are extracted using multiple views of the board, enforcing the orthonormality of the direction cosines matrix columns. An analytical solution is exploited as a first guess for a numerical refinement using a non linear optimization algorithm like the Levenberg-Marquardt algorithm, resulting in a simple and robust method.

Since the aim of this work is to demonstrate the capability of a visual navigation system there is no need to address a more complex camera calibration: a step by step procedure can be found, for example, in [38].

3.5 Way to extract information from a stream of images

Excluding the use of a reference map, the motion is reconstructed from a starting point by processing a stream of sequential images taken by the camera. This implies that the initial conditions are very important and that the navigation system is bound to diverge for error accumulation. Given two consecutive images from the same camera there are two possible way of computing the relative movement: feature tracking (point matching) and optical flow. In a certain sense both methods track points from one image to the other, the first by selecting specified points in both views, the second by computing the variation of whole portions of images.

The first method requires similar feature points to be extracted from both images and then matched: all the points matched in this way are used to determine the rigid transformation between the two views. The quality of the estimation increases with the number of features and matching accuracy process. This procedure is capable of addressing wide motion.

Optical flow is based on the images intensity differentiation to be able to determine the motion. If two images are equal the difference of all pixel brightness gives zeros, but if there is a motion it is possible to determine the pixel position change. The great limitation of such methods is the weakness to large displacement, hence in lunar landing framework is advisable to use the first method [15].

Chapter 4

Features detection and matching

In this Chapter the process of feature extraction and matching is expounded: at first features of the same kind are detected in two consecutive images, then those that appears in both images are matched forming couples. Features can be roughly divided into two subset depending on what they are representing:

- Landmarks (high level features)
- Interesting points (low level features)

The latter includes a lot of different kind of interesting points like corners, edges or even regions. For the sake of brevity in this presentation are shown a corner detector and two region-based detectors.

4.1 Feature detection

In general feature detection algorithms make use of so called *detector*s, particular mathematical operations aimed to find points in an image that have peculiar characteristics. These characteristics are usually exploited confronting the brightness of a point with its neighbors.

4.1.1 Landmarks

Landmarks are feature points that are related to the environment the camera is looking at, implying that some information must be known beforehand. This means that it is possible to program a detector to search for particular objects by using some of their characteristics, like shape for craters on the Moon. The power of such method is that it is possible to match the terrain features to a map in order to determine, at least, the scale and subsequently the full state reconstruction. The drawback is that if no features of this kind are found, the software have to switch to another method. Even if multiple kinds of terrain features are involved, the robustness of the system lies on the probability to find such features. Landmarks are considered high level feature recognized by specific combination of low level features like edges or corners and/or other information, like the direction of the sun. Computation of high level features is in general computationally expensive, therefore less suitable for a fast navigation algorithm.

4.1.2 Corners

There are several corner detectors, but for the sake of brevity only the most popular, the Harris detector, is addressed [36].

Corners are low level features that correspond to a rapid direction changes in edges. If needed, corner points can be exploited to bound shapes and reconstruct high level features. Corner detectors must compute a measure to discriminate corner points from non-corner points, i.e. a measure of curvature. The Harris corner detector exploits image intensity changes to estimate the local curvature of each considered point of coordinates $\{u,\, v\}^T$. The curvature is addressed through the definition of auto-correlation, i.e. a measure of the matching of a signal with itself.

The autocorrelation function in direction $\mathbf{d} = \{x,\, y\}^T$ over a window of size $2w+1$ is defined as

$$\mathbb{E}_{x,y}(u,\, v) = \sum_{i=-w}^{w} \sum_{j=-w}^{w} \left[P_{u+i,v+j} - P_{u+i+x,v+j+y}\right]^2 \tag{4.1}$$

In Moravec's corner detector the measure of curvature is obtained by computing the minimum value of $\mathbb{E}_{x,y}(u,\, v)$ with shifts in the four main directions $\begin{Bmatrix}1 & 0\end{Bmatrix}^T$, $\begin{Bmatrix}0 & 1\end{Bmatrix}^T$, $\begin{Bmatrix}-1 & 0\end{Bmatrix}^T$, $\begin{Bmatrix}0 & -1\end{Bmatrix}^T$. The increment in the image intensity function is approximated by the directional derivative along $\mathbf{d}$ giving the image brightness at the end point as

$$P_{u+i+x,v+j+y} \simeq P_{u+i,v+j} + x\frac{\partial}{\partial u}P_{u+i,v+j} + y\frac{\partial}{\partial v}P_{u+i,v+j} \tag{4.2}$$

Substituting (4.2) in (4.1) and expanding

$$\mathbb{E}_{x,y}(u,\, v) = A(u,\, v)\,x^2 + 2C(u,\, v)\,xy + B(u,\, v)\,y^2 \tag{4.3}$$

$$\mathbb{E}_{x,y}(u,\, v) = \mathbf{d}^T \underbrace{\begin{bmatrix} A(u,\, v) & C(u,\, v) \\ C(u,\, v) & B(u,\, v) \end{bmatrix}}_{\mathbf{M}} \mathbf{d} \tag{4.4}$$

where the three coefficients depends only on the image and the point coordinates as follows.

$$A\left(u,\,v\right) = \sum_{i=-w}^{w}\sum_{j=-w}^{w}\left(\frac{\partial}{\partial u}P_{x+i,y+j}\right)^2 \tag{4.5}$$

$$B\left(u,\,v\right) = \sum_{i=-w}^{w}\sum_{j=-w}^{w}\left(\frac{\partial}{\partial v}P_{x+i,y+j}\right)^2 \tag{4.6}$$

$$C\left(u,\,v\right) = \sum_{i=-w}^{w}\sum_{j=-w}^{w}\left(\frac{\partial}{\partial u}P_{x+i,y+j}\frac{\partial}{\partial v}P_{x+i,y+j}\right) \tag{4.7}$$

The coefficients represent summation of squared components of gradient direction for all the pixel in the window, and in general is weighted through a Gaussian function to reduce the noise sensitivity of the result.

Equation (4.3) has a quadratic form, therefore it has two orthogonal principal axes. It is possible to exploit this property by rotating the function to make the two axis coincide with the principal axis and remove the middle term, that is equivalent of making the matrix **M** diagonal.

$$\mathbb{F}_{x,y}\left(u,\,v\right) = \left(\mathbf{Rd}\right)^T\mathbf{M}\left(\mathbf{Rd}\right) \tag{4.8}$$

$$\mathbb{F}_{x,y}\left(u,\,v\right) = \mathbf{d}^T\underbrace{\mathbf{R}^T\mathbf{M}\mathbf{R}}_{\mathbf{G}}\mathbf{d} \tag{4.9}$$

$$\mathbb{F}_{x,y}\left(u,\,v\right) = \mathbf{d}^T\begin{bmatrix}\alpha & 0\\ 0 & \beta\end{bmatrix}\mathbf{d} \tag{4.10}$$

$$\mathbb{F}_{x,y}\left(u,\,v\right) = \alpha^2x^2 + \beta^2y^2 \tag{4.11}$$

where α and β are point dependent variables proportional to the auto-correlation function along the principal axes. Corner points are characterized by higher values of α and β while if one of the two is significantly smaller than the other the points is a flat border. If both terms are small then the points is not even an edge.

Following this reasoning a curvature measure can be defined as

$$\kappa_k\left(u,\,v\right) = \alpha\beta - k\left(\alpha+\beta\right)^2 \tag{4.12}$$

that, in fact, increases if both values increase accordingly and decreases whenever just one of the two term is large. k is a parameter that selects the sensitivity of the detector: the higher the value the more the measure is noise sensitive. There is no need to actually compute α and β, since the goal is to compute the curvature

$\kappa_k(u, v)$. From linear algebra is known that $\mathbf{G}$ is the orthogonal decomposition of $\mathbf{M}$ representing its eigenvalues.

$$\alpha\beta = A(u, v)\,B(u, v) - C(u, v)^2 \tag{4.13}$$

$$\alpha + \beta = A(u, v) + B(u, v) \tag{4.14}$$

Finally, by substituting (4.13) and (4.14) into (4.12) the curvature measure is estimated.

$$\kappa_k(u, v) = A(u, v)\,B(u, v) - C(u, v)^2 - k\left(A(u, v) + B(u, v)\right)^2 \tag{4.15}$$

Applying this scheme to every point in an image would require an unacceptable computational effort, therefore in practical implementations an edge detector is used before in order to reduce the number of corner candidates [36].

4.1.3 Regions

The purpose of region based feature detection is to solve some of the drawbacks of corner tracking, in particular the sensitivity to image scale. This approach introduces the notion of scale space to find features that have higher probability to be tracked as scale changes.

4.1.3.1 Scale Invariant Feature Transform

The most significant region based feature detection is the Scalar Invariant Feature Transform (SIFT), that has the peculiarity of using a *descriptor* after the detector. Descriptors are mathematical operators that allows the feature to be described within certain parameters, allowing comparison and relevance assessment. SIFT detects features that are invariant to feature size (image scale), rotation and partially to illumination changes. Occlusion and noise effects on feature extraction are reduced [36]. First, the difference of Gaussians operator is applied to an image to identify features of potential interest. The formulation aims to ensure that feature selection does not depend on feature size (scale) or orientation. Features are then analyzed to determine location and scale before the orientation is determined by local gradient direction. Finally features are transformed into a representation that can handle variation in illumination and local shape distortion.

The first step is the computation of the Laplace of Gaussian (LoG) operator obtained by approximation through difference of Gaussians. This process enables the search for second order edges detection.

$$\sigma \nabla^2 g\left(u,\, v,\, \sigma\right) = \frac{\partial g}{\partial \sigma} \simeq \frac{g\left(u,\, v,\, k\sigma\right) - g\left(u,\, v,\, \sigma\right)}{k\sigma - \sigma} \tag{4.16}$$

For the image **P**

$$D\left(u,\, v, \sigma\right) = \left(g\left(u,\, v,\, k\sigma\right) - g\left(u,\, v,\, \sigma\right)\right) * \mathbf{P} \tag{4.17}$$

$$D\left(u,\, v, \sigma\right) = L\left(x,\, y,\, k\sigma\right) - L\left(x,\, y,\, \sigma\right) \tag{4.18}$$

where $g\left(u,\, v,\, \sigma\right)$ is the Gaussian function, $*$ is the convolution operation and $L\left(,\,\right)$ is a scale-space function used to different scale smoothed images definition. Candidate key-points are detected by comparison with the nearest pixel points. The comparison takes place on the scale level, comparing a point and its eight neighbors with the nine points in each of the adjacent scales to determine if it corresponds to a maximum or a minimum. The image can be re-sampled to ensure comparison between different scales.

The key-points candidates are further filtrated rejecting the low local contrast points and the points badly localized along an edge. This procedure requires a uniform thresholding to reject low contrast edges and a curve fitting that address strength, stability and location of such points. Location issues can be addressed by considering the curvature ratio between the edge direction and its perpendicular direction. This can be obtained by thresholding the ratio of the terms in (4.13) and (4.14).

Filtered points are then characterized at each scale. The magnitude M_{SIFT} and orientation ϑ_{SIFT} of the gradient are computed as:

$$M_{SIFT}\left(u,\, v\right) = \sqrt{\left(L\left(x+1,\, y\right) - L\left(x-1,\, y\right)\right)^2 + \left(L\left(x,\, y+1\right) - L\left(x,\, y-1\right)\right)^2} \tag{4.19}$$

$$\vartheta_{SIFT}\left(u,\, v\right) = \arctan\left(\frac{L\left(x,\, y+1\right) - L\left(x,\, y-1\right)}{L\left(x+1,\, y\right) - L\left(x-1,\, y\right)}\right) \tag{4.20}$$

The peak of the orientation histogram of a key-point is selected as the local feature direction and used to derive a canonical orientation. Analyzing regions in the locality of a viewpoint makes the descriptor invariant to rotation as well as reducing viewpoint and non linear changes in brightness sensibility[1] [36]. A complete explanation of SIFT can be found in [39].

[1]Linear changes are removed by the gradient operation

4.1.3.2 Speeded Up Robust Features

The main drawbacks of SIFT are the high computational effort, as stated indirectly in [39], and the number of features delivered [36]. Speeded Up Robust Features (SURF) have been created to cope with this problems as well as maintaining good invariance properties. Instead of using difference of Gaussians SURF employs a second order edge detection approximation at different scales [36,40].

The integral image approach is used to compute the approximation of second order derivatives that in practice are an approximation of the Laplacian of Gaussian (LoG) operator with $\sigma = 1.2$. The image brightness is expanded through Taylor series (4.2) and forms the following Hessian matrix whose maxima are used to derive the features:

$$\mathbf{H} = \begin{bmatrix} L_{xx} & L_{xy} \\ L_{xy} & L_{yy} \end{bmatrix} \tag{4.21}$$

$$L_{xx} = \frac{\partial^2}{\partial x^2} g\left(u,\, v,\, \sigma\right) * P_{x,y} \tag{4.22}$$

$$L_{xy} = \frac{\partial^2}{\partial x \partial y} g\left(u,\, v,\, \sigma\right) * P_{x,y} \tag{4.23}$$

$$L_{yy} = \frac{\partial^2}{\partial y^2} g\left(u,\, v,\, \sigma\right) * P_{x,y} \tag{4.24}$$

the maxima are then derived using the determinant of the Hessian matrix

$$\det\left(\mathbf{H}\right) \simeq L_{xx}L_{yy} - wL_{xy}^2 \tag{4.25}$$

where the parameter w is chosen to balance the components of the equation for better computing.

Scale space is obtained by upscaling the approximation with larger templates, a technique that is faster than the smoothing and re-sampling employed by SIFT and similar methods. Interest points localization over scale is conducted by applying suppression of non maximum in a $3 \times 3 \times 3$ neighborhood, thus considering three different scales. The maxima of the determinant are then interpolated in scale and image space and then orientation-described by vertical and horizontal Haar wavelets.

The wavelets [36] are a rather new application of a rather old principle in signal processing whose main advantage is the allowance of different scale/resolution analysis. Haar wavelets are in substance binary functions that forms averages over set of points and are therefore well suited for image compression. The process consist of successive averaging and differentiating that can be applied at different scales while retaining the local shape. Vertical and horizontal templates in the Haar wavelets can detect points that are brighter in one side with respect to the other side, vertically or horizontally. The direction of gradient can be easily addressed by combining vertical and horizontal wavelet.

SURF operator puts emphasis on speed, performance and optimization by accurate calibration of templates, w factor and interpolation.

This presentation used the rather simple explanation approach found in [36], for a step by step description of the feature extraction algorithm refer to [40, 41].

4.2 Feature Matching

After features detection and extraction from two consecutive images, corresponding couples have to be matched correctly. This kind of procedure falls in the template matching area, therefore it is better to start by explaining the process starting from the matching of a template t within an image $\mathtt{f}$.

4.2.1 Template matching

The distance measure equal to the squared Euclidean distance between a template t and an image $\mathtt{f}$ is

$$d^2_{\mathtt{f},t}(u,\, v) = \sum_{x,\, y} \left[\mathtt{f}(x,\, y) - t(x-u,\, y-v)\right]^2 \tag{4.26}$$

where the summation over x and y is done in the window around the feature stationed in position $\begin{Bmatrix} u & v \end{Bmatrix}^T$. Expanding the squared term gives

$$d^2_{\mathtt{f},t}(u,\, v) = \sum_{x,\, y} \left[\mathtt{f}^2(x,\, y) + t^2(x-u,\, y-v) - 2\mathtt{f}(x,\, y)\, t(x-u,\, y-v)\right] \tag{4.27}$$

The t^2 term is constant and if the image energy $\sum_{x,\, y} \mathtt{f}^2(x,\, y)$ is almost constant, the cross correlation term is

$$c(u,\, v) = \sum_{x,\, y} \mathtt{f}(x,\, y)\, t(x-u,\, y-v) \tag{4.28}$$

If the image energy is not constant then the image correlation may fail. These problems are solved using a normalized cross correlation $\gamma(u,\, v)$ where image and template vectors are normalized.

$$\gamma(u,\, v) = \frac{\sum_{x,\, y} \left[\mathtt{f}(x,\, y) - \bar{\mathtt{f}}_{x,y}\right]\left[t(x-u,\, y-v) - \bar{t}\right]}{\sqrt{\sum_{x,\, y} \left[\mathtt{f}(x,\, y) - \bar{\mathtt{f}}_{x,y}\right]^2 \left[t(x-u,\, y-v) - \bar{t}\right]^2}} \tag{4.29}$$

where $\bar{t}$ is the mean of the template and $\bar{\mathtt{f}}_{x,y}$ is the mean of $\mathtt{f}(x, y)$ in the region under the template [42].

4.2.1.1 Computation

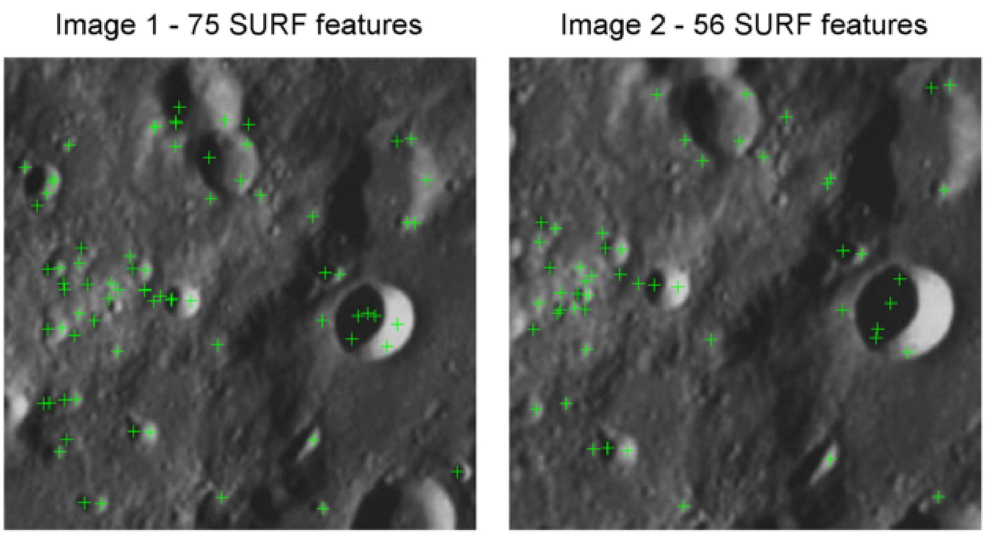
The normalized cross correlation computation (4.29) can be simplified by computing first the numerator. If the search window is of size M^2, the template is N^2 and the images have already been detracted by their mean values the computational cost of (4.29) numerator is about $O\left(2N^2\left(M-N+1\right)^2\right)$, considering additions and multiplications. The computation of the numerator can be carried out by Fourier transform since it is equal to the convolution of the image $\mathtt{f}$ with the inverse template $t(-x, -y)$. With a proper implemented Fast Fourier Transform the computational cost of the numerator is $O\left(30M^2\log_2\left(M\right)\right)$ [43]. The latter method is generally found more performing with $M \simeq N$ and both quite high [42, 43].

4.2.2 From template matching to feature matching

In the case of feature matching it is possible to take both $\mathtt{f}$ and t as portion of an image around a feature point. The matching process of a feature i of image 1 with a feature j in image 2 is simply the computation of the normalized cross correlation $\gamma(u, v)$ between the feature i and all the features detected in image 2: the one with the highest correlation is deemed to be the match feature j. The process is then repeated for all the features in i. With respect to cross correlation the normalized cross correlation reduces the possibilities of having a mismatch, however the possibility of such event is of course not zero. This means that additional algorithms are required to cope with this possibility: pairs that do not represent a coherent motion with the majority of the pairs should be discarded.

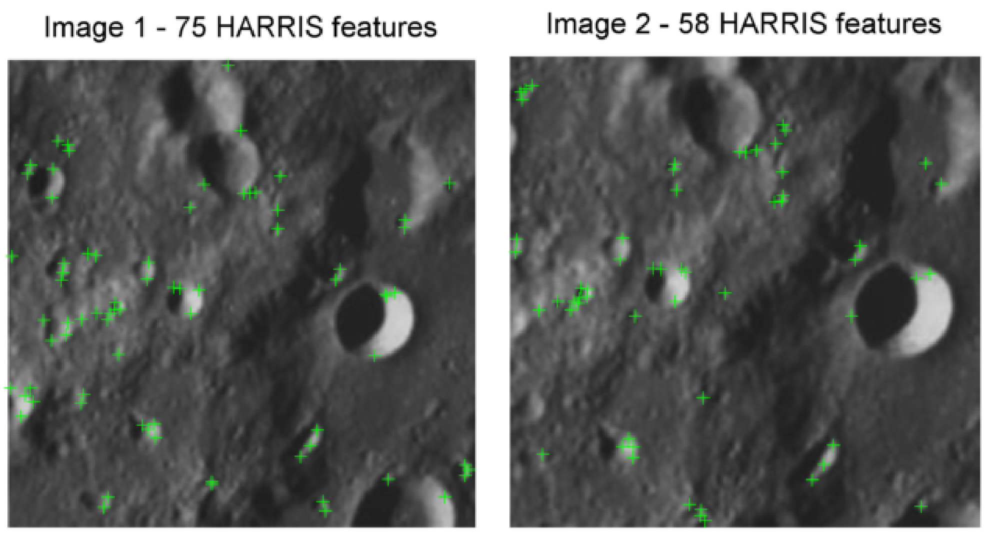
4.3 Feature choice

As stated before it is not convenient to use landmarks like craters in this navigation algorithm in order to retain generality. Between corners, SIFT and SURF the choice fell onto the SURF algorithm, due to its speed and good traceability performance. This can be shown by a simple comparison: Figures 4.1 and 4.2 report the features extracted from two images in a vertical descent by the SURF and Harris detectors. Should be noticed that this is just an example using the computer vision toolbox of Matlab® with default settings for the feature matching in both cases. A more detailed and rigorous analysis require an intensive study and the development of a dedicated matching algorithm.



(a) Image 1 (b) Image 2

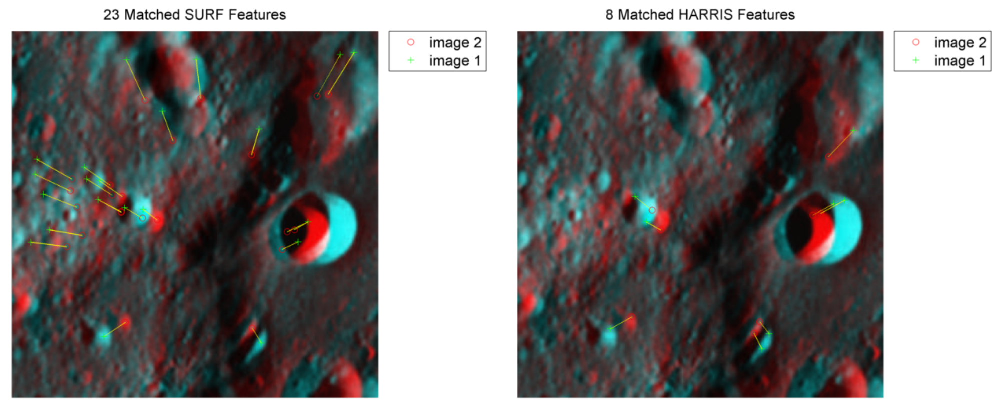
Figure 4.1: SURF features



(a) Image 1 (b) Image 2

Figure 4.2: Harris corner features

In this case the Harris corner detector curvature threshold has been set in order to match the number of features taken in image 1 by SURF detector. The images come from a vertical downward sequence, separated by about 10 seconds. This exaggeration is intentional and aims to make more evident the motion between frames.



(a) SURF (b) Harris corner

Figure 4.3: Matching: SURF vs Harris

Images 4.3 show that with almost the same number of feature extracted from the images, SURF detector delivers more match than Harris corner detector. In other related works the SIFT/SURF have been deemed not necessary [44] while in some other SIFT are presented as a good alternative to crater matching while in occlusion. It is the author opinion that SURF features have adequate computational effectiveness and precision to be used on a Moon lander. It is possible to see SURF as a refinement of corner points, being able to deliver fewer features with a slightly better traceability. The only drawback that has been noticed about SIFT (and SURF) [44] is the weakness to relatively high light variations. That may prove to be problematic when comparing a scene and a map, however in the image stream approach here taken it is less likely to happen since light conditions do not change significantly in a short time.

Chapter 5

Computer Vision algorithms

In this chapter, different algorithms for the rigid motion determination are reported and analyzed. First of all an introduction to motion models for computer vision is addressed, then some algorithms and their computational process are analyzed. Upon such considerations a suitable navigation algorithm approach is proposed.

5.1 Models and properties of image stream

Given two images 1 and 2 it is possible to extract n_1 feature points of pixel camera coordinate $\xi_{1,i} = \begin{Bmatrix} u_1 & v_1 & 1 \end{Bmatrix}_i^T$ from the first and n_2 points $\xi_{2,j} = \begin{Bmatrix} u_2 & v_2 & 1 \end{Bmatrix}_j^T$ from the second. Then a matching procedure is used to get the n features in the first image that have a counterpart in the second one. Of course of the $i \in \mathbb{N} : 1 \leq i \leq n_1$ and $j \in \mathbb{N} : 1 \leq j \leq n_2$ features in the images just a subset is chosen, therefore $n \leq n_1$ and $n \leq n_2$. The different index for each image feature is discarded for the sake of simplicity and the matched couple are referred as $i \in \mathbb{N} : 1 \leq i \leq n$.

5.1.1 Rigid transformation and indetermination

The rigid transformation between two 3-D points $\mathbf{p}_1$ and $\mathbf{p}_2$ is

$$\mathbf{p}_2 = \mathbf{R}_{21}\mathbf{p}_1 + \mathbf{t} \tag{5.1}$$

Where $\mathbf{t}$ is the translation from 1 to 2 and $\mathbf{R}_{21}$ is the rotation that move a vector in the reference frame 1 into the reference frame 2. Assuming that both points are projected onto two different images taken by the same camera

$$\tau_{2,i}\mathbf{K}\xi_{2,i} = \tau_{1,i}\mathbf{R}_{21}\mathbf{K}\xi_{1,i} + \mathbf{t} \tag{5.2}$$

$$\tau_{2,i}\eta_{2,i} = \tau_{1,i}\mathbf{R}_{21}\eta_{1,i} + \mathbf{t} \tag{5.3}$$

if the depths $\tau_{1,i}$ and $\tau_{2,i}$ are known it is possible to determine the rigid transformation or, vice versa, knowing the rigid transformation and determining depths as it is usually done with stereo-cameras. With a single camera the depths are unknown, therefore a multiplication by a scalar does not change the results of the equality: it is possible to normalize the translation vector by dividing for its indeterminable modulus.

$$\frac{\tau_{2,i}}{\|\mathbf{t}\|}\eta_{2,i} = \frac{\tau_{1,i}}{\|\mathbf{t}\|}\mathbf{R}_{21}\eta_{1,i} + \frac{\mathbf{t}}{\|\mathbf{t}\|} \tag{5.4}$$

$$\zeta_{2,i}\mathbf{K}\eta_{2,i} = \zeta_{1,i}\mathbf{R}_{21}\eta_{1,i} + \hat{\mathbf{t}} \tag{5.5}$$

Where $\zeta_{1,i}$ and $\zeta_{2,i}$ are the ratios of depths over translation magnitudes and $\hat{\mathbf{t}}$ is the translation unit vector. In case of an image sequence where image 2 is taken after image 1 the problem can be addressed in two ways: transformation from 1 to 2 or from 2 to 1. The final result is the same but of course the values for $\mathbf{R}_{21}$ and $\hat{\mathbf{t}}$ are going to be different.

The model is the same for all the n features, therefore $\mathbf{R}_{21}$ and $\mathbf{t}$ are collective parameters of the matched features, while each feature depth is a parameter of each single pair.

5.1.2 Epipolar constraint

The so called epipolar constraint [45–47] is a relation between points in a image sequence[1] that must be satisfied due to geometrical reasons. Every couple i must satisfy the epipolar constraint in this form:

$$\xi_{2,i}^T\mathbf{F}\xi_{1,i}\Big|_i = 0 \qquad \forall i \in \mathbb{N} \,:\, 1 \leq i \leq n \tag{5.6}$$

where $\mathbf{F}$ is the fundamental matrix, constant for all matched points between two images. For a couple of feature points i, (5.6) is a scalar equation

[1]or in two images of the same object taken at the same instant by two different cameras. This constraint is often used for stereo vision.

$$u_1u_2F_{11} + u_1v_2F_{21} + u_1F_{31} + v_1u_2F_{12} + v_1v_2F_{22} + v_1F_{32} + u_2F_{13} + v_2F_{23} + F_{33} = 0 \tag{5.7}$$

that can be arranged in a linear system as

$$\begin{bmatrix} u_1u_2 & u_1v_2 & u_1 & v_1u_2 & v_1v_2 & v_1 & u_2 & v_2 & 1 \end{bmatrix} \begin{Bmatrix} F_{11} \\ F_{21} \\ F_{31} \\ F_{12} \\ F_{22} \\ F_{32} \\ F_{13} \\ F_{23} \\ F_{33} \end{Bmatrix} = 0 \tag{5.8}$$

Therefore to determine $\mathbf{F}$ it is possible to write as many lines as the number of couples and solve a least square problem. Details about algorithms that exploit such procedure are given in 5.2.1.

If the intrinsic matrix $\mathbf{K}$ is known, it is possible to address the epipolar constraint using the essential matrix $\mathbf{E}$ instead of $\mathbf{F}$ [46]. Substituting the calibrated coordinates in (5.6)

$$\xi_{2,i}^T \mathbf{F} \xi_{1,i} = 0 \tag{5.9}$$

$$\left(\mathbf{K}^{-1}\eta_{2,i}\right)^T \mathbf{F}\mathbf{K}^{-1}\eta_{1,i} = 0 \tag{5.10}$$

$$\eta_{2,i}^T \underbrace{\mathbf{K}^{-T}\mathbf{F}\mathbf{K}^{-1}}_{\mathbf{E}} \eta_{1,i} = 0 \tag{5.11}$$

In this way it is possible to enforce the epipolar constraint also in homogeneous coordinates. The relation between essential matrix and rigid transformation is of course

$$\mathbf{E} = \mathbf{K}^{-T}\mathbf{F}\mathbf{K}^{-1} \tag{5.12}$$

5.1.2.1 Fundamental and Essential matrix properties

Both $\mathbf{E}$ and $\mathbf{F}$ are semi positive definite and rank deficient. $\mathbf{F}$ has rank 2 due to the intrinsic indetermination in depth and scale factor. In fact the matrix $\mathbf{F}$ can be arbitrary multiplied by a scalar leaving unchanged the results of (5.6): a single-camera sequence alone cannot be used to determine a complete 6 d.o.f rigid transformation. The relation between the essential matrix and the rigid transformation composed by the rotation $\mathbf{R}_{21}$ and the translation direction $\hat{\mathbf{t}}$ [47] is

$$\mathbf{E} = \left[\hat{\mathbf{t}}\times\right]\mathbf{R}_{21} \tag{5.13}$$

This composition inherently contains the rank deficiency. Let us examine the composition of the essential matrix for a small rotation between views

$$\mathbf{E} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \begin{bmatrix} 1 & -\vartheta_z & \vartheta_y \\ \vartheta_z & 1 & -\vartheta_x \\ -\vartheta_y & \vartheta_x & 1 \end{bmatrix}$$

$$\mathbf{E} = \begin{bmatrix} -t_z\vartheta_z - t_y\vartheta_y & -t_z + t_y\vartheta_x & \vartheta_x t_z + t_y \\ t_z + t_x\vartheta_y & -\vartheta_z t_z - t_x\vartheta_x & \vartheta_y t_z - t_x \\ -t_y + t_x\vartheta_z & \vartheta_z t_y + t_x & -\vartheta_y t_y - t_x\vartheta_x \end{bmatrix}$$

It is easy to see that if there is no translation between two views all the essential matrix components should be zero, this implies that the rank of $\mathbf{E}$ is not 2 but zero. This event may also happen when some components are null, for example take the case where just t_x and ϑ_y are non-null[2]

$$\mathbf{E} = \begin{bmatrix} 0 & 0 & 0 \\ t_x\vartheta_y & 0 & -t_x \\ 0 & t_x & 0 \end{bmatrix} \tag{5.14}$$

Any numerical method used to determine such matrix should take care of similar cases.

5.1.2.2 Computing translation and rotation from essential matrix

Once $\mathbf{E}$ has been obtained it is possible to compute both the rotation matrix and the translation direction vector [47]. Through Single Value Decomposition (SVD) it is possible to obtain the left and right matrices $\mathbf{U}$ and $\mathbf{V}$

[2]This implicitly means $t_x \simeq 1$.

$$\mathbf{E} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{5.15}$$

From which can be obtained the rotation matrix multiplying for a matrix $\mathbf{S}$ or its transpose $\mathbf{S}^T$ [47].

$$\mathbf{S} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.16}$$

$$\mathbf{R}_{21} = \begin{cases} \mathbf{U}\mathbf{S}\mathbf{V}^T \\ \mathbf{U}\mathbf{S}^T\mathbf{V}^T \end{cases} \tag{5.17}$$

This means that there are two possible solutions for the rotation and this implies also two distinct translation direction that differs in orientation. Then, there are a total of four different solutions that must be assessed to find the only solution coherent with the motion.

The translation unit vector can be obtained as the eigenvector corresponding to the minimum eigenvalue (theoretically zero) from $\mathbf{U}$. As alternative the following procedure can be used.

$$\mathbf{E}\mathbf{E}^T = \left[\hat{\mathbf{t}}\times\right]\left[\hat{\mathbf{t}}\times\right]^T \tag{5.18}$$

$$\mathbf{E}\mathbf{E}^T = \begin{bmatrix} t_z^2 + t_y^2 & -t_x t_y & -t_z t_x \\ -t_x t_y & t_z^2 + t_x^2 & -t_z t_y \\ -t_z t_x & -t_z t_y & t_x^2 + t_y^2 \end{bmatrix} \tag{5.19}$$

since the translation vector modulus is one

$$t_x^2 + t_y^2 + t_z^2 = 1 \tag{5.20}$$

$$\mathbf{E}\mathbf{E}^T = \begin{bmatrix} 1 - t_x^2 & -t_x t_y & -t_z t_x \\ -t_x t_y & t_z^2 + t_x^2 & -t_z t_y \\ -t_z t_x & -t_z t_y & t_x^2 + t_y^2 \end{bmatrix} \tag{5.21}$$

and since

$$\hat{\mathbf{t}}\hat{\mathbf{t}}^T = \begin{bmatrix} t_x^2 & t_x t_y & t_x t_z \\ t_y t_x & t_y^2 & t_y t_z \\ t_z t_x & t_z, t_y & t_z^2 \end{bmatrix} \tag{5.22}$$

that gives

$$\hat{\mathbf{t}}\hat{\mathbf{t}}^T = \mathbf{I} - \mathbf{E}\mathbf{E}^T \tag{5.23}$$

bust since this requires more computation it is advisable to use the eigenvector method.

5.2 Computer Vision algorithms

There are different numerical methods that can be used to determine the relative motion of the camera using two consecutive images. In this section two algorithms are presented for the fundamental matrix estimation and other three algorithms show the possible application of the 3D point matching.

5.2.1 Fundamental matrix estimation

There are quite a few method to estimate the fundamental matrix (or the essential) with all the features or with a minimum number of these. Among the latter there are the *Five Point Algorithm*, the *Seven Point Algorithm* and the *Eight Point Algorithm*. The first two methods have multiple solution, therefore the number of possible rigid transformation rises. One of the most used methods is the eight point algorithm that make use of just 8 features to determine the fundamental matrix through epipolar constraint between the two views [45].

RAndom SAmple Consensus (RANSAC) based estimators use a random search for the determination of **F**. The method came out in the eighties [48] and many variations have been made through the years.

A deep comparison between a wide number of method for the determination of **F** can be found in [49]. This work is focused on the determination of a robust algorithm, rather than comparing many already-existing algorithms.

5.2.1.1 Eight point algorithm

With n matched points it is possible to construct a simple linear system from (5.8):

$$\mathbf{A}\mathbf{f} = \mathbf{0} \tag{5.24}$$

where $\mathbf{f} = \left\{ F_{11} \quad F_{21} \quad F_{31} \quad F_{12} \quad F_{22} \quad F_{32} \quad F_{13} \quad F_{23} \quad F_{33} \right\}^T$. If $n > 9$ the system is over determined. Limiting to just 9 matches gives a direct solution that can be obtained with a lot of numerical methods for solving linear systems. However, since $\mathbf{F}$ is defined only up to an unknown scale it is needed to enforce also $\|\mathbf{f}\| = 1$. In this way the minimum number of matched point can be reduced to eight, hence the name of the algorithm. Should be noticed that for noiseless measurement also $\mathbf{A}$ is rank deficient. In order to have a non-zero solution in the real case a search for a least square solution minimizing $\|\mathbf{Af}\|$ with constraint $\mathbf{f}^T\mathbf{f} = 1$ must be carried out. Using Lagrange multipliers the solution is the unit eigenvector corresponding to the smallest eigenvalue of $\mathbf{A}^T\mathbf{A}$. Notice that if $\mathbf{A}$ is a 8×9 matrix then $\mathbf{A}^T\mathbf{A}$ is a 9×9 matrix. The eigenpair can be obtained through a single value decomposition or using Jacobi's algorithm. The obtained matrix $\mathbf{F}$ is not rank deficient due to numerical errors and sensor noise, therefore a refinement of $\mathbf{F}$ must be carried out. It is possible to apply a single value decomposition again and generate a matrix $\mathbf{F}'$ such that the minimum eigenvalue (last value of $\mathbf{\Sigma}$) is set to zero. This forceful method is found to minimize the Frobenius norm of $(\mathbf{F}' - \mathbf{F})$.

A more robust estimation can be obtained by the use of normalized coordinates at the price of extra computation. The normalization procedure requires the shifting of the reference system to have the centroid of all features in the center of the image, in order to reduce numerical problems related to the different order of magnitude of the pixel coordinates. Then all the points are scaled such that the average distance from the center is $\sqrt{2}$ (isotropic scaling). All these procedure must be performed independently on each image [45]. Additional methods for further refinement of $\mathbf{F}$, here neglected for the sake of brevity, can be found in [50].

Algorithm 5.1 Eight point algorithm

```
function NormalizedEightPoint($\xi_{1,i}, \xi_{2,j}$)

    isotropic scaling $\leftarrow (\xi_{1,i}, \xi_{2,j})$
    $\mathbf{A} \leftarrow$                                   ▷ Form the matrix
    $\mathbf{U\Sigma V}^T = \mathbf{A}$                        ▷ Solve the system
    $\mathbf{F} \leftarrow \mathbf{f} \leftarrow \mathbf{U}$

    $\mathbf{U\Sigma V}^T = \mathbf{F}$
    $\mathbf{\Sigma}' \leftarrow \mathbf{\Sigma}$              ▷ Null third eigenvalue
    $\mathbf{F}' = \mathbf{U\Sigma}'\mathbf{V}^T$
    reverse isotropic scaling $\leftarrow \mathbf{F}'$

    return $\mathbf{F}$                ▷ Return the transformed fundamental matrix
end function
```

The main limitations of the eight point algorithm are the additional computational load introduced by isotropic scaling and scarce robustness to outliners, i.e. bad feature matches non coherent with the motion.

5.2.1.2 RANSAC

Since the 8-point algorithm is sensitive to noise and to outliners, a RANSAC procedure (presented in its original version, found in [48]) is often used to remove outliners and make use of all the possible tracked features matches. In this presentation of the algorithm the focus is on the original RANSAC in order to describe better its characteristic. The topic is so much explored and popular that has its own "for dummies" version [51]. Given a set S of matched points the RANSAC algorithm flow is described in 5.2.

Algorithm 5.2 RANSAC

```
function RANSAC($\xi_{1,i}, \xi_{2,i}$)

    while $k < n_{max}$ do                                      ▷ Continue until max iteration
        $i$ = RandomNumber (8)
        $\mathbf{F}_k \leftarrow$ NormalizedEightPoint ($\xi_{1,i}, \xi_{2,i}$)
        $\mathbf{r}_k \leftarrow$                                ▷ Get the residual of all point in $S$
        $C_k \leftarrow \mathbf{r}_k$                            ▷ Get the consensus set

        if $C_k > C_{\text{best}}$ then                          ▷ Check if iteration is better
            $C_{\text{best}} \leftarrow C_k$
            $\mathbf{F}_{\text{best}} \leftarrow \mathbf{F}_k$
            if $C_{\text{best}} > C_{\text{min}}$ then           ▷ Stop if consensus is enough
                break
            end if
        end if

    end while

    return $\mathbf{F}_{\text{best}}$
end function
```

The minimum number of iterations can be addressed knowing w, the fraction of inliners in S. If N points are randomly selected from S:

- The probability to take an inliner are approximately w^N ,
- The probability that not all N points are inliners is given by $1 - w^N$,
- The probability that not all N points are inliners in k iterations is $\left(1 - w^N\right)^k$,
- The probability that in k iteration, at least once, all N points are inliners is $p_{success} = 1 - \left(1 - w^N\right)^k$.

Solving for k gives the inequality

$$k > \frac{\log\left(1 - p_{success}\right)}{\log\left(1 - w^N\right)} \tag{5.25}$$

this is an optimistic estimation since also inliners can produce bad results due to noise or numerical errors.

Numerous variations of this algorithm have been proposed, among the which the MSAC [52] that sorts the pairs with regard of their fitness to the model. Inliners are sorted basing on their fitness while the outliners have a constant consensus. There are various kind of RANSAC algorithms that try to reduce the computational burden, while others try to increase the precision of the results with methods that include M-estimators [53] or Iteratively Reweighted Least Squares combined with Levenberg-Marquardt optimization [50].

The drawback of this method is that requires a lot of computation to get the determination of the essential matrix and a good result is not always delivered. It is, however, robust to noise and outliners.

5.2.2 Non-linear estimation of rigid transformation

The aforementioned methods try to find **F** to get rotation and translation direction, however there are multiple solutions and the determination of translation magnitude needs to be addressed anyway. Instead of relying on the estimation of essential matrix it is possible to make use of a rigid transformation model to estimate directly and without multiple solutions the transformation between the two views. If the system is linear then the least squares framework can still be applied, however the non linear projective model and the rotation parametrization with quaternions, chosen to prevent singularity, require non linear estimation.

In literature the most popular method to solve such problems applied to computer vision is the Levenberg-Marquard (LM) algorithm, a non linear minimization that can shift between two other methods: the gradient descent algorithm and the Newton-Gauss algorithm. These methods can often be seen as a non linear extension of the least-squares problem, and similarly the robustness of the estimation can be affected by noise and, most importantly, by outliners. An outliner removal routine can be introduced inside the LM algorithm to eliminate the wrong matches without reducing the precision of the estimation nor causing divergence [15].

5.2.2.1 Problem formulation

In order to describe the application of the aforementioned algorithms to computer vision, here they are firstly presented in the more easy-to-get curve fitting problems.

In certain conditions these methods can be used as curve fitting algorithms also for computer vision. The dissertation here presented follows the same scheme of [54].

Assume to have a fitting function $\hat{\mathbf{y}}\left(\mathbf{a},\, t_i\right)$ of independent variables t_i and a vector $\mathbf{a}$ of m parameters that have to fit n measured points y_i corresponding to the independent variables t_i. The sum of weighted residuals between the measured data and the fitting function is then minimized with respect the fitting parameters vector $\mathbf{a}$. This scalar value is often called the chi-squared error criterion:

$$\chi^2\left(\mathbf{a}\right) = \sum_{i=1}^{n} \left(\frac{y_i - \hat{y}_i\left(\mathbf{a},\, t_i\right)}{w_i}\right)^2 \tag{5.26}$$

In matrix formulation[3]

$$\chi^2\left(\mathbf{a}\right) = \mathbf{y}^T\mathbf{W}\mathbf{y} - 2\mathbf{y}^T\mathbf{W}\hat{\mathbf{y}} + \hat{\mathbf{y}}^T\mathbf{W}\hat{\mathbf{y}} \tag{5.27}$$

where $\mathbf{W}$ is the $m \times m$ diagonal weighting matrix with $W_{ii} = {}^{1}\!/_{w_i^2}$, $\mathbf{y}$ and $\hat{\mathbf{y}}$ are vectors containing all the n measures and their fits. w_i can be considered a measure of the goodness of the measure, however in computer vision is not always easy to address such quantities. In the practical implementation of the algorithm this aspect is not investigated, here it is reported for the sake of generality.

5.2.2.2 Gradient descent method

The gradient descent method, often called steepest descent method, minimize $\chi^2\left(\mathbf{a}\right)$ by updating the parameter $\mathbf{a}$ in the opposite direction of the gradient of $\chi^2\left(\mathbf{a}\right)$. This means that the parameter vector $\mathbf{a}$ tends to go in the opposite direction of maximum error increment, therefore going in the right direction to reduce it. For simple problems this algorithm is found be highly converging towards the minimum of the fitting function. The gradient of the objective function is

$$\frac{\partial \chi^2\left(\mathbf{a}\right)}{\partial \mathbf{a}} = -\left(\mathbf{y} - \hat{\mathbf{y}}\right)^T \mathbf{W}\mathbf{J} \tag{5.28}$$

where $\mathbf{J}$ is the $m \times n$ Jacobian of the fitting function, defined as

$$\mathbf{J} = \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{a}} \tag{5.29}$$

The parameter variation is

$$\delta\mathbf{a} = \alpha\mathbf{J}^T\mathbf{W}\left(\mathbf{y} - \hat{\mathbf{y}}\right) \tag{5.30}$$

[3]since $\mathbf{y}^T\mathbf{W}\hat{\mathbf{y}}$ is a scalar and $\mathbf{W}$ is diagonal the use of $\mathbf{y}^T\mathbf{W}\hat{\mathbf{y}} = \hat{\mathbf{y}}^T\mathbf{W}\mathbf{y}$ is allowed.

where α is the positive scalar that determines the length of the step that has to be taken opposite to the gradient direction. At each step k the Jacobian $\mathbf{J}_k$, residual $\mathbf{r}_k = (\mathbf{y} - \hat{\mathbf{y}}_k)$ and the parameter variation $\delta\mathbf{a}_k$ are computed. Then the parameter variation is computed and the parameter vector updated

$$\mathbf{a}_{k+1} = \mathbf{a}_k + \delta\mathbf{a}_k \tag{5.31}$$

Should be noticed that the Jacobian can be computed through numerical methods or using the analytical expression if the fitting function is well known.

5.2.2.3 Newton-Gauss method

Unlike the previous method, in this algorithm the objective function is considered approximately quadratic in the parameters when the solution is near the minimum [55]. The fitting function can be approximated through Taylor series expansion as follows

$$\hat{\mathbf{y}}\left(\mathbf{a} + \delta\mathbf{a}\right) = \hat{\mathbf{y}}\left(\mathbf{a}\right) + \frac{\partial\hat{\mathbf{y}}}{\partial\mathbf{a}}\delta\mathbf{a} + O\left(\|\delta\mathbf{a}|^2\right) \tag{5.32}$$

$$\hat{\mathbf{y}}\left(\mathbf{a} + \delta\mathbf{a}\right) \simeq \hat{\mathbf{y}}\left(\mathbf{a}\right) + \mathbf{J}\delta\mathbf{a} \tag{5.33}$$

substituting in the squared criterion for the perturbed parameter vector, an additional contribution due to the perturbation appears

$$\chi^2\left(\mathbf{a} + \delta\mathbf{a}\right) = \chi^2\left(\mathbf{a}\right) - 2\left(\mathbf{y} - \hat{\mathbf{y}}\right)^T \mathbf{W}\mathbf{J}\delta\mathbf{a} + \delta\mathbf{a}^T\mathbf{J}^T\mathbf{W}\mathbf{J}\delta\mathbf{a} \tag{5.34}$$

The gradient in the perturbation is

$$\frac{\partial\chi^2\left(\mathbf{a} + \delta\mathbf{a}\right)}{\partial\delta\mathbf{a}} = \frac{\partial}{\partial\delta\mathbf{a}}\left(\chi^2\left(\mathbf{a}\right) - 2\left(\mathbf{y} - \hat{\mathbf{y}}\right)^T \mathbf{W}\mathbf{J}\delta\mathbf{a} + \delta\mathbf{a}^T\mathbf{J}^T\mathbf{W}\mathbf{J}\delta\mathbf{a}\right) \tag{5.35}$$

$$\frac{\partial\chi^2\left(\mathbf{a} + \delta\mathbf{a}\right)}{\partial\delta\mathbf{a}} = -2\left(\mathbf{y} - \hat{\mathbf{y}}\right)^T \mathbf{W}\mathbf{J} + 2\delta\mathbf{a}^T\mathbf{J}^T\mathbf{W}\mathbf{J} \tag{5.36}$$

At the minimum the value of such gradient should be zero, therefore the equation for the perturbation vector can be written as

$$\left[\mathbf{J}^T\mathbf{W}\mathbf{J}\right]\delta\mathbf{a} = \mathbf{J}^T\mathbf{W}\left(\mathbf{y} - \hat{\mathbf{y}}\right) \tag{5.37}$$

The update is carried out as in the gradient descent method.

5.2.2.4 Levenberg-Marquard method

The original formulation of the Levenberg method adaptively varies the parameter update between the two methods previously discussed, by the introduction of the parameter λ [56].

$$\left[\mathbf{J}^T\mathbf{W}\mathbf{J} + \lambda\mathbf{I}\right]\delta\mathbf{a} = \mathbf{J}^T\mathbf{W}\left(\mathbf{y} - \hat{\mathbf{y}}\right) \tag{5.38}$$

As $\lambda \to 0$ the method approaches the Newton-Gauss method, while for higher values of λ it approaches the steepest descent method. In fact, if $\lambda \gg \text{diag}\left\{\mathbf{J}^T\mathbf{W}\mathbf{J}\right\}$ or the problem is dominated by the diagonal elements of $\left[\mathbf{J}^T\mathbf{W}\mathbf{J} + \lambda\mathbf{I}\right]$, it is possible to approximate as

$$\delta\mathbf{a} \simeq \underbrace{\frac{1}{\lambda}}_{\alpha}\mathbf{J}^T\mathbf{W}\left(\mathbf{y} - \hat{\mathbf{y}}\right) \tag{5.39}$$

Marquard, while resuming Levenberg work, suggested in [57] the use of local parameter variation curvature to speed up the computation and have a better matrix conditioning, leading to the well known form

$$\left[\mathbf{J}^T\mathbf{W}\mathbf{J} + \lambda\text{diag}\left\{\mathbf{J}^T\mathbf{W}\mathbf{J}\right\}\right]\delta\mathbf{a} = \mathbf{J}^T\mathbf{W}\left(\mathbf{y} - \hat{\mathbf{y}}\right) \tag{5.40}$$

At each iteration the solution is checked with a metric function to see if the step is pushing towards the minimization of $\chi^2\left(\right)$ and how fast it is. The metric function of the step k is defined as

$$\rho\left(\delta\mathbf{a}_k\right) = \frac{\chi^2\left(\mathbf{a}_k\right) - \chi^2\left(\mathbf{a}_k + \delta\mathbf{a}_k\right)}{2\delta\mathbf{a}_k^T\left(\lambda_k\delta\mathbf{a}_k + \mathbf{J}^T\mathbf{W}\left(\mathbf{y} - \hat{\mathbf{y}}\left(\mathbf{a}_k\right)\right)\right)} \tag{5.41}$$

If the metric of the step is above a user specified positive threshold ϵ_1 then the step is deemed worthy, $\mathbf{a}$ is updated as $\mathbf{a}_{k+1} = \mathbf{a}_k + \delta\mathbf{a}_k$ and λ is decreased. Otherwise the step is not accepted, the parameters not updated and λ is increased to set the algorithm more on the gradient direction. This should always happen when $\chi^2\left(\mathbf{a}_k + \delta\mathbf{a}\right) > \chi^2\left(\mathbf{a}_k\right)$, meaning that the algorithm is getting away from the minimum. There are different updating procedures but the most common is

$$\begin{cases} \lambda_k = \frac{\lambda_{k-1}}{v_\downarrow} & \text{if } \rho\left(\delta\mathbf{a}_k\right) > \epsilon_1 \\ \lambda_k = \lambda_{k-1}\cdot v_\uparrow & \text{if } \rho\left(\delta\mathbf{a}_k\right) < \epsilon_1 \end{cases} \tag{5.42}$$

where the updating parameters $v_\uparrow$ and $v_\downarrow$ can be different. The choice of these parameters influences the speed of the LM algorithm.

The minimum is considered to be reached if one of the following criteria is satisfied:

- gradient convergence, when the maximum component of the gradient is below a user defined threshold,
- parameter convergence, when the maximum of $|{}^{\delta \mathbf{a}_k}/_{\mathbf{a}_k}|$ is below a user defined threshold,
- fitting convergence, when $\frac{\chi^2(\mathbf{a}_k)}{n-m+1}$ is below a user defined threshold,
- maximum number of iterations reached.

The LM algorithm, as well as the other two methods, can be used to minimize the sum of squared error of a generic function $f\left(\mathbf{a},\, t_i\right) = 0$, in fact

$$\left[\mathbf{J}^T\mathbf{W}\mathbf{J} + \lambda \mathrm{diag}\left\{\mathbf{J}^T\mathbf{W}\mathbf{J}\right\}\right]\delta\mathbf{a} = -\mathbf{J}^T\mathbf{W}\mathbf{r} \tag{5.43}$$

where the residual is the $n \times 1$ vector whose components are defined as

$$r_i = f\left(\mathbf{a},\, t_i\right) \tag{5.44}$$

It is possible to address the covariance matrix $\mathbf{V}_a$ of the estimated parameters in order to assess the quality of the parameter determination:

$$\mathbf{V}_a = \left(\mathbf{J}^T\mathbf{W}\mathbf{J}\right)^{-1} \tag{5.45}$$

5.2.2.5 3-D fitting

In case of a 3-D fitting there are two options to use this method: have the residual and Jacobian length equal to $3n$ or address a scalar function that includes all the 3-D components. In the first case the problem lies on the metric and fitting parameter since they became vectors, while on the second it is possible to fall into numerical problems. For example to minimize the distance between two 3-D points i and j one must write

$$d_i = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2 + \left(z_i - z_j\right)^2} \tag{5.46}$$

$$\frac{\partial d_{ij}}{\partial x_j} = -\frac{1}{d_{ij}}\left(x_i - x_j\right) \tag{5.47}$$

the Jacobian components values increase as the distance is shortened, therefore when $d_i \to 0$ the values of the Jacobian tends to infinity and the solution diverges. This can be settled in two ways: using the squared distances for the minimization or solve the perturbation equation using a SVD and eliminating the rows and column that makes the solution divergent. Needless to say that the first way is computationally safer and more effective.

5.2.2.6 Drawbacks of the non-linear methods

All the above non-linear methods have the same weakness of least-squares problems: noise and outliners. In the computer vision problem outliners are the most dangerous source of errors and in some cases it is not easy to solve. One strategy could be the use of a RANSAC procedure to determine the outliners before using the non-linear methods, however this increase greatly the computation time, therefore a removal strategy inside the algorithm is the best approach, although not easy.

For instance if the parameter state is feature dependent the removal of outliners should be handled with extra care. In other cases, an outliner removal strategy must be used without a user defined absolute threshold but using a relative threshold in order to avoid exaggerated point reductions. The strategy from [15] has been deemed very effective and thus adopted for this algorithm. Given the residual $\mathbf{r}$ of the n features, the removal procedure needs to

Algorithm 5.3 Outliners removal

```
function OutlinerRemoval($\mathbf{r}$)

    $\mathbf{r}_\uparrow \leftarrow \mathbf{r}$                                                        ▷ Sort $\mathbf{r}$ in ascending order
    $\epsilon = \mathbf{r}_\uparrow\left(\frac{n}{2}\right) + 3\left[\mathbf{r}_\uparrow\left(\frac{3n}{4}\right) - \mathbf{r}_\uparrow\left(\frac{n}{4}\right)\right]$      ▷ Set the relative threshold

    for $i \leftarrow 1, n$ do
        if $r_\uparrow(i) < \epsilon$ then
            ISINLINER $\leftarrow i$
        end if
    end for

    return ISINLINER                                              ▷ Return the inliner couples
end function
```

This strategy is effective and remove the needs of RANSAC costly computation to have a robust estimation.

5.2.3 Comparison between numerical methods

The epipolar-based algorithms have the intrinsic limitation derived by the fundamental matrix handling:

- multiple solutions,

- inability to determine translation magnitude by themselves,
- error-prone in case of null translation.

Moreover RANSAC-based routines are in general quite costly for the result that they can provide. This implies that the non linear methods and in particular the LM algorithm are chosen for the computation of the lander state. This choice is also supported by the good results of [15].

Chapter 6

Navigation algorithm

In order to determine the 6 d.o.f. of the lander during the descent phase without using other sensors, it is almost mandatory to use a non-linear method like the Levenberg-Marquard algorithm. The motion parameters and a rigid body transformation that relates two different views have to be defined in order to use such method. From [15] the full state reconstruction is possible if each feature depth is known: with a single camera this is impossible. However, if the depth of all the feature in the first image are known, it is possible to determine the depths of feature in the second and have a real 3D reconstruction, eliminating the indetermination in scale. This imposes a requirement for the algorithm: the initial distance from the ground must be known beforehand (with other sensors).

For each feature i the relation between the first and the second image features is given by (5.3), $\mathbf{R}_{12}$ is parametrized in terms of the quaternion $\mathbf{q}$. In this expression for sure $\eta_{2,i}$ and $\eta_{1,i}$ are known, while all $\tau_{1,i}$ must be addressed by knowing the previous camera position w.r.t. the terrain. Following this model the LM algorithm parameter state vector is thus

$$\mathbf{a} = \begin{Bmatrix} \mathbf{t} & \mathbf{q} & \tau_{2,1} & \cdots & \tau_{2,i} & \cdots & \tau_{2,n} \end{Bmatrix}^T \tag{6.1}$$

that is a $7+n$ long vector. This suggest that the computational burden is quite heavy and feature dependent. In order to address the problem using the LM algorithm it is better to re-write (5.3) as a 3D point matching. To better understand the problem a reference frame change is performed. The variables are rotated in the the terrain reference frame using the rotation matrix $\mathbf{R}_{tc,1}$ that rotates from the camera position at instant 1 to the terrain frame.

$$\tau_{1,i}\eta_{1,i} = \tau_{2,i}\mathbf{R}_{12}\eta_{2,i} + \mathbf{t} \tag{6.2}$$

$$\mathbf{R}_{tc,1}\tau_{1,i}\eta_{1,i} = \tau_{2,i}\left[\mathbf{R}_{tc,1}\mathbf{R}_{12}\right]\eta_{2,i} + \left[\mathbf{R}_{tc,1}\mathbf{t}\right] \tag{6.3}$$

$$\tau_{1,i}\mathbf{R}_{tc,1}\eta_{1,i} = \tau_{2,i}\mathbf{R}_{tc,2}\eta_{2,i} + \mathbf{t}_t \tag{6.4}$$

$$\tau_{1,i}\mathbf{R}_{tc,1}\eta_{1,i} = \tau_{2,i}\mathbf{R}_{tc,2}\eta_{2,i} + \left[\mathbf{x}_{l,2} - \mathbf{x}_{l,1}\right] \tag{6.5}$$

$$\tau_{1,i}\mathbf{R}_{tc,1}\eta_{1,i} + \mathbf{x}_{l,1} = \tau_{2,i}\mathbf{R}_{tc,2}\eta_{2,i} + \mathbf{x}_{l,2} \tag{6.6}$$

Equation (6.6) represent the matching of the features in the terrain reference frame. It uses the lander position at instant 1 $\mathbf{x}_{l,1}$, the 3D points $(\tau_{1,i}\eta_{1,i})$ reconstructed by the camera and of course the rotation from the camera reference frame to the terrain reference frame. The same model is thus applied to data of instant 2. The 3D terrain position of a generic feature i detected by the camera at an instant k is thus

$$\mathbf{p}_{k,i} = \tau_{k,i}\mathbf{R}_{tc,k}\eta_{k,i} + \mathbf{x}_{l,k} \tag{6.7}$$

In this reference frame, the lander position vector at the second time instant is considered instead of the translation between views in LM optimization:

$$\mathbf{a} = \begin{Bmatrix} \mathbf{x}_{l,2} & \mathbf{q}_{1,2} & \tau_{2,1} & \cdots & \tau_{2,i} & \cdots & \tau_{2,n} \end{Bmatrix}^T \tag{6.8}$$

6.1 Planar model

Before searching for ways to simplify the computation, the determination of all $\tau_{1,i}$ has to be settled. In navigation problems where the camera is looking at the terrain from an high altitude it is possible to simplify greatly the computation by approximating the terrain surface as a flat plane. If the standard deviation of the terrain altitude is negligible w.r.t. the altitude of the lander, errors due to this approximation should be small. Moreover the terrain is considered still since on the Moon atmospheric elements and life forms are not present.

Taking a plane described in vector notation as a set of points $\mathbf{p}_j$ with origin $\mathbf{p}_0$ and normal direction $\hat{\mathbf{n}}_{pl}$

$$(\mathbf{p}_j - \mathbf{p}_0) \cdot \hat{\mathbf{n}}_{pl} = 0 \tag{6.9}$$

and a line described by a origin point $\mathbf{l}_0$, its direction $\hat{\mathbf{l}}$ and the distance from the origin d_k

$$\mathbf{p}_k = d_k\hat{\mathbf{l}} + \mathbf{l}_0 \tag{6.10}$$

The distance of intersection between line origin and plane can be obtained easily as follows, if all quantities are expressed in the same reference frame.

$$\left(d_k\hat{\mathbf{l}} + \mathbf{l}_0 - \mathbf{p}_0\right) \cdot \hat{\mathbf{n}}_{pl} = 0 \tag{6.11}$$

$$(\mathbf{l}_0 - \mathbf{p}_0) \cdot \hat{\mathbf{n}}_{pl} = -d_k\hat{\mathbf{l}} \cdot \hat{\mathbf{n}}_{pl} \tag{6.12}$$

$$d_k = -\frac{(\mathbf{l}_0 - \mathbf{p}_0) \cdot \hat{\mathbf{n}}_{pl}}{\hat{\mathbf{l}} \cdot \hat{\mathbf{n}}_{pl}} \tag{6.13}$$

Taking the terrain surface as a flat plane with the origin of axes coinciding with the reference frame, i.e. $\mathbf{p}_0 = \mathbf{0}$, and $\mathbf{l}_0$ as the camera position vector, the term $\mathbf{l}_0 \cdot \hat{\mathbf{n}}_{pl}$ is the distance of the camera from the terrain: the lander altitude. The term $\hat{\mathbf{l}} \cdot \hat{\mathbf{n}}_{pl}$ then relates the orientation of the line w.r.t. the terrain. The line used is no less than the line that connect the feature on the camera with its corresponding real 3D point:

$$\left\{\hat{\mathbf{l}}_{1,i}\right\} = \frac{\eta_{1,i}}{\|\eta_{1,i}\|} \tag{6.14}$$

however the feature direction components are given in the camera reference frame, therefore this vector has to be rotated in the terrain relative frame. The rotation $\mathbf{R}_{tc,1}$ that changes a vector from the camera reference to a terrain reference frame at instant 1 is used as follows:

$$\hat{\mathbf{l}}_{1,i} = \mathbf{R}_{tc,1}\frac{\eta_{1,i}}{\|\eta_{1,i}\|} \tag{6.15}$$

Therefore the depth of feature in the first image, given the lander altitude when the first shot has been taken Z_1, is

$$\tau_{1,i} = -\frac{Z_1}{\left(\mathbf{R}_{tc,1}\frac{\eta_{1,i}}{\|\eta_{1,i}\|}\right) \cdot \begin{Bmatrix} 0 \\ 0 \\ 1 \end{Bmatrix}} \tag{6.16}$$

where the plane normal has been taken as the local zenith direction. Following this thread, the 3D points of the first image in the terrain reference frame is determined by substituting (6.16) into (6.7):

$$\mathbf{p}_{1,i} = -\frac{Z_1}{\left(\mathbf{R}_{tc,1}\frac{\eta_{1,i}}{\|\eta_{1,i}\|}\right)\cdot\begin{Bmatrix}0\\0\\1\end{Bmatrix}}\mathbf{R}_{tc,1}\eta_{1,i} + \begin{Bmatrix}X_1\\Y_1\\Z_1\end{Bmatrix} \tag{6.17}$$

In a similar fashion

$$\mathbf{p}_{2,i} = -\frac{Z_2}{\left(\mathbf{R}_{tc,2}\frac{\eta_{2,i}}{\|\eta_{2,i}\|}\right)\cdot\begin{Bmatrix}0\\0\\1\end{Bmatrix}}\mathbf{R}_{tc,2}\eta_{2,i} + \begin{Bmatrix}X_2\\Y_2\\Z_2\end{Bmatrix} \tag{6.18}$$

Since each point in the terrain frame must coincide in both picture

$$-\frac{Z_1}{\left(\mathbf{R}_{tc,1}\frac{\eta_{1,i}}{\|\eta_{1,i}\|}\right)\cdot\begin{Bmatrix}0\\0\\1\end{Bmatrix}}\mathbf{R}_{tc,1}\eta_{1,i} + \begin{Bmatrix}X_1\\Y_1\\Z_1\end{Bmatrix} = -\frac{Z_2}{\left(\mathbf{R}_{tc,2}\frac{\eta_{2,i}}{\|\eta_{2,i}\|}\right)\cdot\begin{Bmatrix}0\\0\\1\end{Bmatrix}}\mathbf{R}_{tc,2}\eta_{2,i} + \begin{Bmatrix}X_2\\Y_2\\Z_2\end{Bmatrix} \tag{6.19}$$

This expression is basically (6.16) substituted into (6.6). In the end the choice in which reference frame the model is written is up to the user. From here onwards terrain coordinates are considered, gaining a better understanding of the problem.

The planar approximation, within its applicability limits, also enables to greatly reduce the computational burden by not including each feature depth in the minimization. The LM parameter state vector now is

$$\mathbf{a} = \begin{Bmatrix}\mathbf{x}_{l,2} & \mathbf{q}_{1,2}\end{Bmatrix}^T \tag{6.20}$$

6.2 Derivation of Jacobian

In the LM algorithm the knowledge of the model allows to compute the Jacobian directly. Jacobian depends, of course, on the function that the LM algorithm is going to minimize, however given the general 3D point formulation is possible to address the partial derivatives independently.

6.2.1 Partial derivatives

By expanding the model of (6.7) for the i feature in the second of a couple of images, all the optimization variables of (6.20) are visible:

$$\mathbf{p}_{2,i} = -\frac{Z_2}{\left(\mathbf{R}_{tc,1}\mathbf{R}_{12}\frac{\eta_{2,i}}{\|\eta_{2,i}\|}\right)\cdot\begin{Bmatrix}0\\0\\1\end{Bmatrix}}\mathbf{R}_{tc,1}\mathbf{R}_{12}\eta_{2,i} + \begin{Bmatrix}X_2\\Y_2\\Z_2\end{Bmatrix} \tag{6.21}$$

Instead of $\mathbf{t}$ now there is the camera position in the terrain reference frame, therefore the partial derivatives w.r.t. the lander position are

$$\frac{\partial}{\partial X_2}\mathbf{p}_{2,i} = \begin{Bmatrix}1\\0\\0\end{Bmatrix} \tag{6.22}$$

$$\frac{\partial}{\partial Y_2}\mathbf{p}_{2,i} = \begin{Bmatrix}0\\1\\0\end{Bmatrix} \tag{6.23}$$

$$\frac{\partial}{\partial Z_2}\mathbf{p}_{2,i} = \begin{Bmatrix}0\\0\\1\end{Bmatrix} - \frac{1}{\left(\mathbf{R}_{tc,1}\mathbf{R}_{12}\frac{\eta_{2,i}}{\|\eta_{2,i}\|}\right)\cdot\begin{Bmatrix}0\\0\\1\end{Bmatrix}}\mathbf{R}_{tc,1}\mathbf{R}_{12}\eta_{2,i} \tag{6.24}$$

The rotation is expressed using the quaternion $\mathbf{q} = \begin{Bmatrix}q_1 & q_2 & q_3 & q_4\end{Bmatrix}$ connected to the Euler angle θ and Euler axis $\begin{Bmatrix}x_e & y_e & z_e\end{Bmatrix}$ as follows:

$$\mathbf{q} = \begin{Bmatrix}x_e\sin\theta & y_e\sin\theta & z_e\sin\theta & \cos\theta\end{Bmatrix} \tag{6.25}$$

The Direction Cosine Matrix (DCM) can be expressed in terms of quaternion components

$$\mathbf{R}_{12} = \begin{bmatrix} q_1^2-q_2^2-q_3^2+q_4^2 & 2\left(q_1q_2+q_3q_4\right) & 2\left(q_1q_3-q_2q_4\right) \\ 2\left(q_1q_2-q_3q_4\right) & -q_1^2+q_2^2-q_3^2+q_4^2 & 2\left(q_2q_3+q_1q_4\right) \\ 2\left(q_1q_3+q_2q_4\right) & 2\left(q_2q_3-q_1q_4\right) & -q_1^2-q_2^2+q_3^2+q_4^2 \end{bmatrix} \tag{6.26}$$

then the partial derivatives of the DCM w.r.t. the quaternion components are

$$\mathbf{Q}_1 = \frac{\partial}{\partial q_1}\mathbf{R}_{12} = 2\begin{bmatrix} q_1 & q_2 & q_3 \\ q_2 & -q_1 & q_4 \\ q_3 & -q_4 & -q_1 \end{bmatrix} \tag{6.27}$$

$$\mathbf{Q}_2 = \frac{\partial}{\partial q_2}\mathbf{R}_{12} = 2\begin{bmatrix} -q_2 & q_1 & -q_4 \\ q_1 & q_2 & q_3 \\ q_4 & q_3 & -q_2 \end{bmatrix} \tag{6.28}$$

$$\mathbf{Q}_3 = \frac{\partial}{\partial q_3}\mathbf{R}_{12} = 2\begin{bmatrix} -q_3 & q_4 & q_1 \\ -q_4 & -q_3 & q_2 \\ q_1 & q_2 & q_3 \end{bmatrix} \tag{6.29}$$

$$\mathbf{Q}_4 = \frac{\partial}{\partial q_4}\mathbf{R}_{12} = 2\begin{bmatrix} q_4 & q_3 & -q_2 \\ -q_3 & q_4 & q_1 \\ q_2 & -q_1 & q_4 \end{bmatrix} \tag{6.30}$$

Therefore the partial derivatives of $\mathbf{p}_{2,i}$ with respect to the quaternion can be addressed considering $\hat{\mathbf{n}}_{pl} = \begin{Bmatrix} 0 & 0 & 1 \end{Bmatrix}^T$ and $\hat{\mathbf{l}}_{2,i} = \mathbf{R}_{tc,2}\frac{\eta_{2,i}}{\|\eta_{2,i}\|}$ for simplicity.

$$\frac{\partial}{\partial q_1}\mathbf{p}_{2,i} = -\frac{Z_2\mathbf{R}_{tc,1}}{\hat{\mathbf{l}}_{2,i}\cdot\hat{\mathbf{n}}_{pl}}\left(\mathbf{Q}_1 - \mathbf{R}_{12}\frac{1}{\hat{\mathbf{l}}_{2,i}\cdot\hat{\mathbf{n}}_{pl}}\left(\left(\mathbf{R}_{tc,1}\mathbf{Q}_1\frac{\eta_{2,i}}{\|\eta_{2,i}\|}\right)\cdot\hat{\mathbf{n}}_{pl}\right)\right)\eta_{2,i} \tag{6.31}$$

$$\frac{\partial}{\partial q_2}\mathbf{p}_{2,i} = -\frac{Z_2\mathbf{R}_{tc,1}}{\hat{\mathbf{l}}_{2,i}\cdot\hat{\mathbf{n}}_{pl}}\left(\mathbf{Q}_2 - \mathbf{R}_{12}\frac{1}{\hat{\mathbf{l}}_{2,i}\cdot\hat{\mathbf{n}}_{pl}}\left(\left(\mathbf{R}_{tc,1}\mathbf{Q}_2\frac{\eta_{2,i}}{\|\eta_{2,i}\|}\right)\cdot\hat{\mathbf{n}}_{pl}\right)\right)\eta_{2,i} \tag{6.32}$$

$$\frac{\partial}{\partial q_3}\mathbf{p}_{2,i} = -\frac{Z_2\mathbf{R}_{tc,1}}{\hat{\mathbf{l}}_{2,i}\cdot\hat{\mathbf{n}}_{pl}}\left(\mathbf{Q}_3 - \mathbf{R}_{12}\frac{1}{\hat{\mathbf{l}}_{2,i}\cdot\hat{\mathbf{n}}_{pl}}\left(\left(\mathbf{R}_{tc,1}\mathbf{Q}_3\frac{\eta_{2,i}}{\|\eta_{2,i}\|}\right)\cdot\hat{\mathbf{n}}_{pl}\right)\right)\eta_{2,i} \tag{6.33}$$

$$\frac{\partial}{\partial q_4}\mathbf{p}_{2,i} = -\frac{Z_2\mathbf{R}_{tc,1}}{\hat{\mathbf{l}}_{2,i}\cdot\hat{\mathbf{n}}_{pl}}\left(\mathbf{Q}_4 - \mathbf{R}_{12}\frac{1}{\hat{\mathbf{l}}_{2,i}\cdot\hat{\mathbf{n}}_{pl}}\left(\left(\mathbf{R}_{tc,1}\mathbf{Q}_4\frac{\eta_{2,i}}{\|\eta_{2,i}\|}\right)\cdot\hat{\mathbf{n}}_{pl}\right)\right)\eta_{2,i} \tag{6.34}$$

6.2.2 Jacobian

Two different models can be considered for the representation of the Jacobian in the LM algorithm. The first can be considered the 3D extension of the fitting problem exploited to explain the LM algorithm, defining the fitting function as simply equal to (6.21). In this case the Jacobian is composed of $3n$ rows as follows:

$$\mathbf{J}_i = \frac{\partial \mathbf{p}_{2,i}}{\partial \mathbf{a}} \tag{6.35}$$

$$\mathbf{J}_i = \begin{bmatrix} \frac{\partial}{\partial X_2}\mathbf{p}_{2,i} & \frac{\partial}{\partial Y_2}\mathbf{p}_{2,i} & \frac{\partial}{\partial Z_2}\mathbf{p}_{2,i} & \frac{\partial}{\partial q_1}\mathbf{p}_{2,i} & \frac{\partial}{\partial q_2}\mathbf{p}_{2,i} & \frac{\partial}{\partial q_3}\mathbf{p}_{2,i} & \frac{\partial}{\partial q_4}\mathbf{p}_{2,i} \end{bmatrix} \tag{6.36}$$

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_1 \\ \vdots \\ \mathbf{J}_i \\ \vdots \\ \mathbf{J}_n \end{bmatrix} \tag{6.37}$$

This formulation, as stated before, has the weakness of having fitting parameters and metric as vectors, increasing the difficulty of handling the confirmation of a step towards the minimum.

The second model involves the minimization of a function that describes the error in a scalar way. The simplest choice is the distance between the reconstructed point $\mathbf{p}_{2,i} = \begin{Bmatrix} x_2 & y_2 & z_2 \end{Bmatrix}_i^T$ and the 3D points of the previous image $\mathbf{p}_{1,i} = \begin{Bmatrix} x_1 & y_1 & z_1 \end{Bmatrix}_i^T$

$$f(\mathbf{a},\, \eta_{2,i}) = \left. \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \right|_i \tag{6.38}$$

To address the Jacobian it is needed the derivation w.r.t. the coordinates, for example

$$\frac{\partial}{\partial x_2} f(\mathbf{a},\, \eta_{2,i}) = \left. \frac{(x_1 - x_2)}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}} \right|_i \tag{6.39}$$

as predicted this function tends to infinity as the error approaches zero, therefore is not suited for this problem unless special care is taken in the linear system computation. A more suitable variation is

$$f(\mathbf{a},\, \eta_{2,i}) = \left. (x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2 \right|_i \tag{6.40}$$

where the derivatives are proportional to the error in components, in fact

$$\frac{\partial}{\partial x_2} f(\mathbf{a}, \eta_{2,i}) = 2(x_1 - x_2)|_i \tag{6.41}$$

$$\frac{\partial}{\partial y_2} f(\mathbf{a}, \eta_{2,i}) = 2(y_1 - y_2)|_i \tag{6.42}$$

$$\frac{\partial}{\partial z_2} f(\mathbf{a}, \eta_{2,i}) = 2(z_1 - z_2)|_i \tag{6.43}$$

$$\frac{\partial}{\partial \mathbf{p}_{2,i}} f(\mathbf{a}, \eta_{2,i}) = 2(\mathbf{p}_{1,i} - \mathbf{p}_{2,i})^T \tag{6.44}$$

This leads to

$$\mathbf{J}_i = \frac{\partial}{\partial \mathbf{a}} f(\mathbf{a}, \eta_{2,i}) \tag{6.45}$$

$$\mathbf{J}_i = \frac{\partial f(\mathbf{a}, \eta_{2,i})}{\partial \mathbf{p}_{2,i}} \begin{bmatrix} \frac{\partial \mathbf{p}_{2,i}}{\partial X_2} & \frac{\partial \mathbf{p}_{2,i}}{\partial Y_2} & \frac{\partial \mathbf{p}_{2,i}}{\partial Z_2} & \frac{\partial \mathbf{p}_{2,i}}{\partial q_1} & \frac{\partial \mathbf{p}_{2,i}}{\partial q_2} & \frac{\partial \mathbf{p}_{2,i}}{\partial q_3} & \frac{\partial \mathbf{p}_{2,i}}{\partial q_4} \end{bmatrix} \tag{6.46}$$

$$\mathbf{J}_i = 2(\mathbf{p}_{1,i} - \mathbf{p}_{2,i})^T \begin{bmatrix} \frac{\partial \mathbf{p}_{2,i}}{\partial X_2} & \frac{\partial \mathbf{p}_{2,i}}{\partial Y_2} & \frac{\partial \mathbf{p}_{2,i}}{\partial Z_2} & \frac{\partial \mathbf{p}_{2,i}}{\partial q_1} & \frac{\partial \mathbf{p}_{2,i}}{\partial q_2} & \frac{\partial \mathbf{p}_{2,i}}{\partial q_3} & \frac{\partial \mathbf{p}_{2,i}}{\partial q_4} \end{bmatrix} \tag{6.47}$$

The Jacobian row has dimension 1×7 since $\frac{\partial f(\mathbf{a}, \eta_{2,i})}{\partial \mathbf{p}_{2,i}}$ has dimension 1×3 and the terms in brackets are the same as in (6.36). With this approach the Jacobian total dimension is reduced to $n \times 7$ and the problem is scalar, easing its handling.

6.2.3 Alternate formulation

It is possible to show that the quaternion parametrization has a weakness in case of small rotations. Take (6.34) and a case where the camera is pointing downwards. For small rotations ϑ_x, ϑ_y and ϑ_z the matrix $\mathbf{Q}_4$ is simplified:

$$\mathbf{R}_{12} \simeq \begin{bmatrix} 1 & -\vartheta_z & \vartheta_y \\ \vartheta_z & 1 & -\vartheta_x \\ -\vartheta_y & \vartheta_x & 1 \end{bmatrix} \tag{6.48}$$

$$\mathbf{Q}_4 \simeq 2 \begin{bmatrix} 1 & \frac{\vartheta_z}{2} & -\frac{\vartheta_y}{2} \\ -\frac{\vartheta_z}{2} & 1 & \frac{\vartheta_x}{2} \\ \frac{\vartheta_y}{2} & -\frac{\vartheta_x}{2} & 1 \end{bmatrix} \tag{6.49}$$

$$\mathbf{Q}_4 \simeq \mathbf{I} + \mathbf{R}_{12}^T \tag{6.50}$$

With the camera pointing downwards from an high altitude, ignoring misalignment on the direction parallel to the terrain

$$\eta_{2,i} \simeq \begin{Bmatrix} 0 & 0 & 1 \end{Bmatrix}^T \tag{6.51}$$

$$\mathbf{R}_{tc,1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{6.52}$$

Substituting into (6.34)

$$\frac{\partial}{\partial q_4}\mathbf{p}_{2,i} \simeq Z_2 \begin{Bmatrix} -3\vartheta_y \\ 3\vartheta_x \\ 0 \end{Bmatrix} \tag{6.53}$$

while for other components it gets, for example

$$\frac{\partial}{\partial q_1}\mathbf{p}_{2,i} \simeq 2Z_2 \begin{Bmatrix} \vartheta_z + \vartheta_x\vartheta_y \\ 1 - \vartheta_x^2 \\ 0 \end{Bmatrix} \tag{6.54}$$

With small angles approximation the second order terms can be removed

$$\frac{\partial}{\partial q_1}\mathbf{p}_{2,i} \simeq Z_2 \begin{Bmatrix} 2\vartheta_z \\ 2 \\ 0 \end{Bmatrix} \tag{6.55}$$

It is straightforward to see that the order of magnitude of some elements is very different: in fact the middle terms in (6.53) and (6.54) are bound to $\vartheta_x \ll 1$. Sometimes this can result in a bad conditioned matrix for the LM that badly affects the result of the algorithm. This can be interpreted as a result of "saturation" of one quaternion component and since the rotation between two views cannot be too high is unlikely that the first three components would cause problems. Solving directly for $\mathbf{R}_{tc,2}$ is an exception to this reasoning.

Without invoking particular checks on the Jacobian and linear system resolution it is possible, looking at (6.25), to use a quaternion parametrization limiting the Euler angle within 90°. This can be done removing the fourth component of the quaternion from the minimization parameters and substituting the unitary requirement of the quaternion in the rotation parametrization as follows

$$\overline{q_4} = \sqrt{1 - \left(q_1^2 + q_2^2 + q_3^2\right)} \tag{6.56}$$

$$\mathbf{R}_{12} = \begin{bmatrix} 1 - 2q_2^2 - 2q_3^2 & 2\left(q_1q_2 + q_3\overline{q_4}\right) & 2\left(q_1q_3 - q_2\overline{q_4}\right) \\ 2\left(q_1q_2 - q_3\overline{q_4}\right) & 1 - 2q_1^2 - 2q_3^2 & 2\left(q_2q_3 + q_1\overline{q_4}\right) \\ 2\left(q_1q_3 + q_2\overline{q_4}\right) & 2\left(q_2q_3 - q_1\overline{q_4}\right) & 1 - q_1^2 - q_2^2 \end{bmatrix} \tag{6.57}$$

therefore the matrices $\mathbf{Q}_1$, $\mathbf{Q}_2$ and $\mathbf{Q}_3$ are modified accordingly. The partial derivative of q_4 w.r.t. the other three components are

$$\frac{\partial q_4}{\partial q_1} = \frac{q_1}{q_4} \tag{6.58}$$

$$\frac{\partial q_4}{\partial q_2} = \frac{q_2}{q_4} \tag{6.59}$$

$$\frac{\partial q_4}{\partial q_3} = \frac{q_3}{q_4} \tag{6.60}$$

That leads to

$$\mathbf{Q}_4 = \begin{bmatrix} 0 & q_3 & -q_2 \\ -q_3 & 0 & q_1 \\ q_2 & q_1 & 0 \end{bmatrix} \tag{6.61}$$

$$\mathbf{Q}_1 = \frac{\partial}{\partial q_1}\mathbf{R}_{12} = 2\begin{bmatrix} 0 & q_2 & q_3 \\ q_2 & -2q_1 & q_4 \\ q_3 & -q_4 & -2q_1 \end{bmatrix} + \frac{q_1}{q_4}\mathbf{Q}_4 \tag{6.62}$$

$$\mathbf{Q}_2 = \frac{\partial}{\partial q_2}\mathbf{R}_{12} = 2\begin{bmatrix} -2q_2 & q_1 & -q_4 \\ q_1 & 0 & q_3 \\ q_4 & q_3 & -2q_2 \end{bmatrix} + \frac{q_2}{q_4}\mathbf{Q}_4 \tag{6.63}$$

$$\mathbf{Q}_3 = \frac{\partial}{\partial q_3}\mathbf{R}_{12} = 2\begin{bmatrix} -2q_3 & q_4 & q_1 \\ -q_4 & -2q_3 & q_2 \\ q_1 & q_2 & 0 \end{bmatrix} + \frac{q_3}{q_4}\mathbf{Q}_4 \tag{6.64}$$

The form of equations (6.31), (6.32) and (6.33) is still maintained by simply changing the definition of $\mathbf{Q}_1$, $\mathbf{Q}_2$ and $\mathbf{Q}_3$. This strategy enables numerical robustness and eliminates one variable from the state vector $\mathbf{a}$, reducing the computation time. The only limitation is that in this formulation q_4 is always positive and since it is defined as the cosine of the Euler angle of the rotation this limits the Euler angle between ±90°. It is an acceptable limitation since the lander it is not supposed to turn abruptly by 90° between two consecutive frames.

6.3 Weakness assessment

The algorithm presented may have an important weakness caused by the enforced simplifications. Since it is assumed that the surface the camera views is planar there is a possibility that the vision system cannot distinguish out of plane rotations from parallel translation. When using a reference map or stereo vision this may not happen since 3D points have a relative altitude that may help the vision system to distinguish a rotation from a translation. 3D information are not recoverable from single camera view alone, the use of a gyroscope might be the best solution.

Gyroscopes are sensors capable of sensing inertial angular velocities and can be used to determine rotation though proper integration. In lunar landing scenarios gyroscopes measures must be handled with care since the celestial body is rotating. Even if the rotation of the Moon is small compared with the descent time window, it is always present and became a source of errors if not counted in the attitude reconstruction. Using gyroscopes to determine rotation while the vision system only determines translation is a possible solution. The system results to be faster but prone to attitude drift and less precise. Such a system is very similar to an INS system, however the drift in position should be way inferior.

6.4 Navigation Algorithm: Dokuganryuu

Taking into account the numerical methods and limitations of the navigation sensors considered, the final version of the navigation algorithm here presented exploits gyroscopes to reconstruct rotations and the camera addresses the translation. The LM algorithm is used to determine the lander position based on the previous position, the images of the previous and the current instant and of course the rotation computed by the gyroscopes.

The name of the algorithm here presented is Dokuganryuu, in honor to the Japanese feudal lord Date Masamune, main sponsor of the city of Sendai where the experimental activity to validate such algorithm has been carried out. Dokuganryuu means «*one-eyed dragon*» in Japanese and was the nickname of Date Masamune because his right eye was blind since childhood, making this name befitting the navigation system itself.

6.4.1 Algorithm flow

The algorithm has been devised in different steps, given a previous image $\mathbf{P}_k$ and the current image $\mathbf{P}_{k+1}$, respectively taken at time t_k and t_{k+1} the algorithm does:

Algorithm 6.1 Dokuganryuu navigation algorithm

```
procedure DOKUGANRYUU

    loop

        $\omega_m \leftarrow$                                        ▷ Get gyro measure
        $\mathbf{R}_{tc,k+1} \leftarrow (\mathbf{R}_{tc,k}, \omega_m)$      ▷ Get attitude from gyroscope measure

        $\mathbf{P}_k, \mathbf{P}_{k+1} \leftarrow$                          ▷ Take images
        $(\xi_{k+1,i}, \xi_{k,j}) \leftarrow (\mathbf{P}_k, \mathbf{P}_{k+1})$    ▷ Detect SURF
        Match feature $i, j$                                 ▷ Match features to get couples
        $(\eta_{k,i}, \eta_{k+1,i}) \leftarrow (\xi_{k,i}, \xi_{k+1,i})$      ▷ Transform coordinates with $\mathbf{K}$
        $\mathbf{p}_k \leftarrow (\mathbf{x}_{l,k}, \mathbf{R}_{tc,k}, \eta_{k,i})$      ▷ Obtain 3D point of first image

        $\mathbf{x}_{l,k+1} \leftarrow$ LevenbergMarquardt $(\eta_{k+1,i}, \mathbf{p}_k, \mathbf{R}_{tc,k}, \mathbf{R}_{tc,k+1}, \mathbf{x}_{l,k})$
        $k \leftarrow k + 1$

        return $(\mathbf{R}_{tc,k+1}, \mathbf{x}_{l,k+1})$
    end loop

end procedure
```

6.4.2 Levenberg-Marquard implementation

The LM algorithm takes the 3D points reconstructed from the previous image and an initial guess that can be either the old position or the forecast position based on previous ones. In order to have an initial evaluation the residual and fitting function are computed using the initial guess. The first value of $\chi^2(\mathbf{a}_0)$ is used as a reference in the metric computation while the residual is used to roughly remove the worst outliners using 5.3:

The LM algorithm requires to set λ_0, $v_\uparrow$, $v_\downarrow$ in some way before the actual computation. There is not a univocal solution for the choice of these parameters, different authors have worked on the topic [54, 58, 59] but in general λ_0 is found to depend on the distance of the initial guess from the final solution. Basically, the further the initial guess is supposed to be from the final solution the higher λ should be. A bigger λ_0 is always suggested, even if this could reduce the speed of the algorithm.

Algorithm 6.2 Levenberg-Marquard for planar matching

```
function LevenbergMarquard($\eta_{2,i}, \mathbf{p}_1, \mathbf{R}_{tc,1}, \mathbf{R}_{tc,2}, \mathbf{a}_0$)

    $\mathbf{r}_0 \leftarrow (\eta_{2,}, \mathbf{p}_1, \mathbf{R}_{tc,1}, \mathbf{R}_{tc,2}, \mathbf{a}_0)$
    $\chi^2(\mathbf{a}_0) \leftarrow \mathbf{r}_0$
    outliner removal $\{\mathbf{r}_0\}$                                ▷ Gross outliner removal

    while $k \neq n_{max}$ do                                    ▷ Continue until max iterations

        $\mathbf{J}_k, \mathbf{r}_k \leftarrow (\eta_{2,}, \mathbf{p}_1, \mathbf{R}_{tc,1}, \mathbf{R}_{tc,2}, \mathbf{a}_k)$
        $\delta\mathbf{a}_k \leftarrow \left[\mathbf{J}_k^T\mathbf{J}_k + \lambda_k \mathrm{diag}\left\{\mathbf{J}_k^T\mathbf{J}_k\right\}\right]\delta\mathbf{a}_k = -\mathbf{J}_k^T\mathbf{r_k}$
        $\rho(\delta\mathbf{a}_k + \mathbf{a}_k) \leftarrow \left(\chi^2(\mathbf{a}_k), \chi^2(\delta\mathbf{a}_k + \mathbf{a}_k)\right)$

        if $\rho(\delta\mathbf{a}_k + \mathbf{a}_k) > \epsilon_1$ then                       ▷ Check the metric

            $\mathbf{a}_{k+1} \leftarrow \mathbf{a}_k + \delta\mathbf{a}_k$                    ▷ Update the state parameter vector

            if max $\left(\mathbf{J}_k^T\mathbf{r}_k\right) < \epsilon_2$ then                 ▷ Check for gradient convergence
                break
            else if max $\left|\frac{\delta\mathbf{a}_k}{\mathbf{a}_k}\right| < \epsilon_3$ then             ▷ Check for parameter convergence
                break
            else if $\frac{\chi^2(\mathbf{a}_{k+1})}{n-m+1} < \epsilon_4$ then               ▷ Check for fitting convergence
                break
            end if

            outliner removal $\{\mathbf{r}_k\}$
            $\lambda_{k+1} = \frac{\lambda_k}{v_\downarrow}$                                    ▷ $\lambda$ update
            $count \leftarrow 0$

        else
            $\lambda_{k+1} = \lambda_k \cdot v_\uparrow$                                  ▷ $\lambda$ update
            $count \leftarrow count + 1$

            if $count > n_{fail}$ then                         ▷ Check for consecutive failure
                break
            end if

        end if

    end while

    return a                                           ▷ Return the best estimate
end function
```

Chapter 7

Experimental validation

In order to verify the quality of the designed algorithm, as well as to identify its limits, an experimental activity has been carried out. The main purpose of this activity has been the demonstration of the potential of the algorithm. Despite the limited time and resources, the results are enough to demonstrate the potential of such algorithm.

7.1 Experimental setup

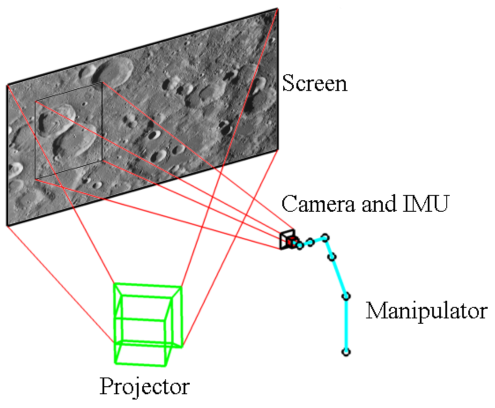


Figure 7.1: Experimental setup sketch

The experimental setup designed for the validation of the navigation algorithm is composed by:

- navigation suite (camera and IMU)
- robotic manipulator

- projector of lunar surface
- screen

An example of the disposition of these elements is presented in Figure 7.1. Camera and IMU are fixed together on an aluminum plate connected to the robotic arm tip, the connection is designed to be flexible to allow different camera orientations. The screen with the lunar surface projected images is positioned about 2 meters from the camera and illuminated by a projector, placed on a table beside the robotic arm in order to avoid projection interference, as shown in the figure. Lights were kept to a minimum in order to have a good projected image on the screen. The aforementioned elements of the setup are presented in details.

7.1.1 Camera

The camera selected for this experiment is the Matrix Vision BlueFox-MLC205. Camera technical specifications are summarized in Table7.1.

Table 7.1: Camera data sheet

Parameter	**Value**	**Units**
Resolution	2592×1944	pixels
Max Frame rate	5.8	Hz
Shutter type	Rolling/Global Reset	
Sensor size	1/2.5”	
Pixel size	2.2×2.2	µm
Exposure time	$10 \div 10^6$	µs
ADC resolution/output	$10 / 10 \div 8$	bit
SNR	> 38	dB
DR (normal / HDR)	> 70	dB
Sensor Manufacturer	Aptina	
Sensor Name	MT9P031	

Frame rate and resolution have been reduced during the tests. In fact current space camera system cannot afford such high performance. Also, the actual computational load depends depends on the size of the image stream. If the image processing algorithm is not fast enough having an high frame rate is just useless.

The framerate for the following tests has been fixed at 2 fps and images are cropped to form a 300×300 pixel stream of images.

7.1.2 IMU

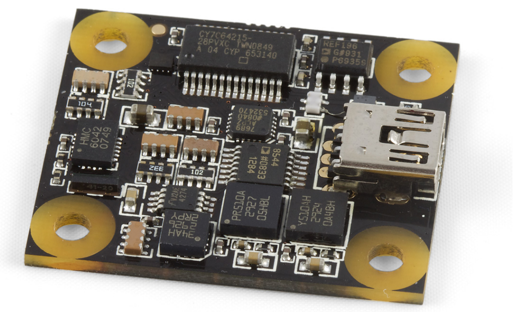


Figure 7.2: Phidget IMU

The Phidgets Spatial 3/3/3 1056 version 100 (product release) has been selected as IMU. It includes a 3-axis accelerometer, a 3-axis gyroscope and a 3-axis magnetic compass (not considered in this research project). Table 7.2 shows related technical data.

Before the test campaign, the sensor has been tested for noise, drift and bias when standing still. It has been found that the measured acceleration is slightly different from the standard $9.81\,{}^{m}/{}_{s^2}$; a small bias has been spotted also in angular rate measures around all three axes. Then, a IMU calibration procedure has been performed independently before each navigation system test runs, after experimental activity.

Please notice that the sensor always takes samples at the higher frequency, regardless of the user requests. The actual output measure is computed internally by averaging all the samples taken between two consecutive user request. This means that sampling at $8\,ms$ produces an output signal that is the mean between two consecutive samplings at $4\,ms$. In order to reconstruct the weak signal of the IMU the acquisition system has been set to sample at the maximum allowed speed.

Table 7.2: IMU Data sheet

(a) Accelerometer

Parameter	Value	Units
Resolution	228	µg
Bandwidth	110	Hz
Max Acceleration	±5	g
X-Axis Noise Level	300	µg
Y-Axis Noise Level	300	µg
Z-Axis Noise Level	500	µg
Rotation Error	2	mg

(b) Gyroscope

Parameter	Value	Units
Max Angular Speed	400	°/s
Resolution	0.02	°/s
Drift	4	°/min

(c) Board

Parameter	Value	Units
Max Current Consumption	45	mA
Min Sampling Speed	1	s
Max Sampling Speed	4	ms
Analog to Digital Converter Resolution	16	bit
USB Min Voltage	4.8	V
DC USB Max Voltage	5.3	V
Min Operating Temperature	0	°C
Max Operating Temperature	70	°C

7.1.2.1 Synchronization

IMU and camera take samples independently, but some form of synchronization is required to properly match the signals of the two sensors. In this first application, data are saved together at the time instant they are written, with the PC internal temporal marker. This provide a good synchronization for post processing data, but in case of a real system the timing should be obtained in a more rigorous and controlled way.

7.1.3 Manipulator

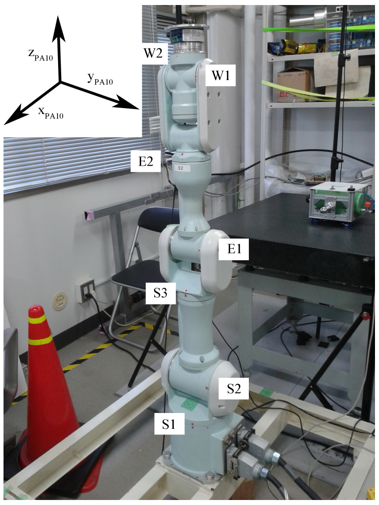


Figure 7.3: Mitsubishi PA10 manipulator

The manipulator used for the experiments is the 7 joints PA10 from Mitsubishi. Its joints permit it to have a good spatial excursion and a good rotational movement of the tip where the navigation system is attached. Manipulator data are reported in Table 7.3, while joints identification and reference frame are reported on Figure 7.3.

Table 7.3: Manipulator Data sheet

(a) General properties

Parameter	Value	Units
Shoulder reach	315	mm
Upper arm	450	mm
Lower arm	500	mm
Wrist reach	80	mm
Max integrated speed	1.55	m/s
Load Capacity	10	kgf
Maximum reach	1345	mm

(b) Joints

Joint	Limit angle	Max Speed
S1	±177°	±57.3°/s
S2	±91°	±57.3°/s
S3	±174°	±114.6°/s
E1	±137°	±114.6°/s
E2	±255°	±360°/s
W1	±165°	±360°/s
W2	±360°	±360°/s

In order to fix the navigation system to the PA10 tip an adapter made using aluminum plates and aluminum profiles has been designed and realized. The sensor assembly can be mounted pointing toward any orthogonal direction.

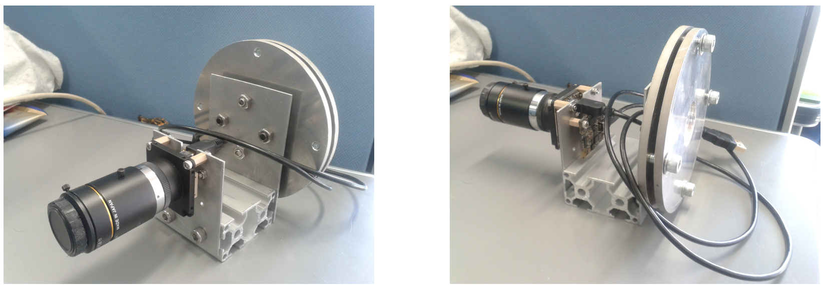


Figure 7.4: System with adapter

The system exchange data with the acquisition PC through two 5m long USB cables.

7.1.3.1 Integration

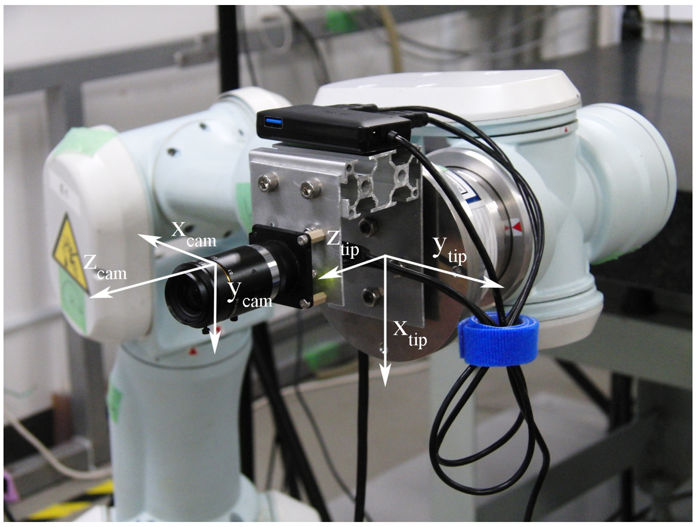


Figure 7.5: Integration with the PA10

Figure 7.5 shows one of the integration configurations between navigation hardware and manipulator arm. This configuration has been used for all the motion where the camera moves towards the lunar surface.

In the sequences that exploit a translation parallel to the ground the robotic arm starting point was symmetric w.r.t. the one in Figure 7.5. That configuration cannot introduce a pitching rotation, therefore for one particular sequence the positioning has been devised differently. During those simulations the camera is not positioned as in Figure 7.5 but it exploits the flexibility of the connections, leading to a different rotation between PA10 data and camera coordinates.

In all simulations a coordinate transformation of PA10 data is performed in order to have the reference in terrain coordinates. The rotation between the tip coordinates and the camera coordinates has been exploited in order to determine the initial attitude conditions of the navigation system w.r.t. terrain coordinates.

7.1.3.2 Motion

The robotic arm can be moved using three different programs:

- point to point by joint angle command,
- point to point by position command,
- tip velocity command.

Point to point by joint is the simplest method, but requires the user to solve an inverse dynamic problem in order to convert the desired trajectory into joint angles profiles. Mitsubishi provided a position control where the manipulator can be controlled by position and orientation of the tip: in this case the inverse dynamic is directly solved by robotic arm's software. Singularities problems can occur during simulation. Moreover, in both point to point control modes a tremble movement, arising at the beginning of each motion, has been observed and detected by the IMU: unrealistic peaks contaminated the signal.

This lead to the choice of tip velocity control. In this mode the PA10 internal software solves the inverse dynamics and provides encoder measures for all the joints during the whole run. Even in this mode singularity may compromise the whole sequence, therefore the sequences used in the following tests have been obtained by trial and error procedure. In order to distinguish IMU signal from its noise the tip velocity have been set not too slow and the maximum excursion of the tip results to be confined due to geometrical constraints.

7.1.4 Simulated lunar surface

A 3D mock-up of real lunar terrain is costly and difficult to obtain; moreover there are some general problems in the camera focus regulation due to limited space in the laboratory. In fact, in the case of a real landing scenario, the distance from the ground during the navigation phase is in the order of km and is shortened just on the final phase. With such distance the camera can be set with a focus at ∞ in order to see everything clearly. A surface put on the ground would have been too close to the camera due to the manipulator arm reach, therefore it has been decided to use a vertical surface placed at about 2 meters from the PA10.

The best method found is to use a projector and with lunar-like surfaces on a screen. This requires a good planning of lights during the experiments, however it is the fastest and cheapest way to address a preliminary testing of the algorithm.

The main drawback in this approach is the lack of dimensionality of projected images. Since the navigation system, in its computation, assumes the surface to be flat the experimental validation could incur in inconsistency. Further experiments with 3D models should be considered in future activities.

Three different lunar surface images[1], depicted in Figures 7.6, 7.7 and 7.8, have been considered in the test campaign. The first image is to be referred as Crater Field (CF), the second to Crater Field Overexposed (CFO) and the third to Lunar Plains (LP). The use of lunar images has been decided in order to have experimental data for future testing: any image could have been used by the navigation algorithm since Dokuganryuu do not exploit high level features like craters. These three images have been selected among others also for their size and resolution in order to have good projections on the screen.

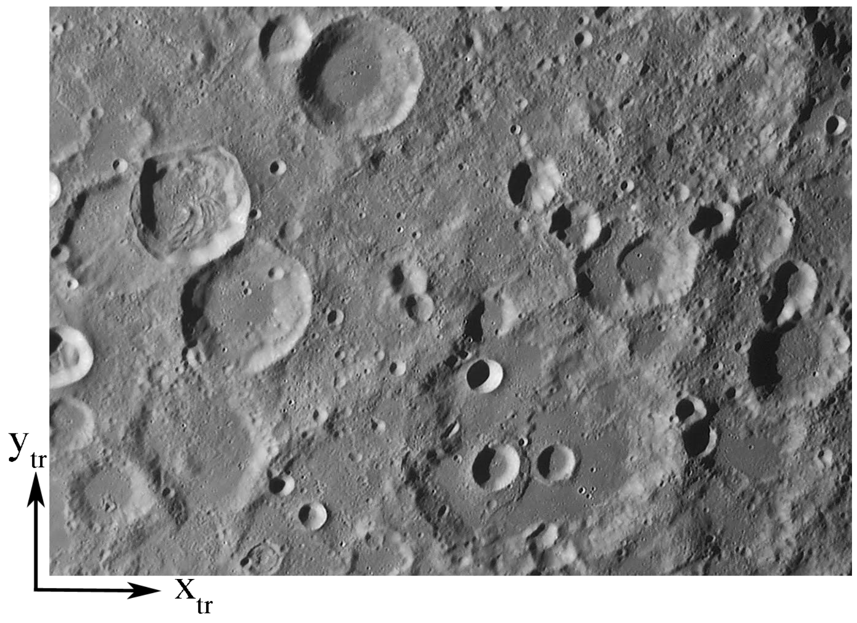


Figure 7.6: Crater Field

[1]Freely taken from the internet.

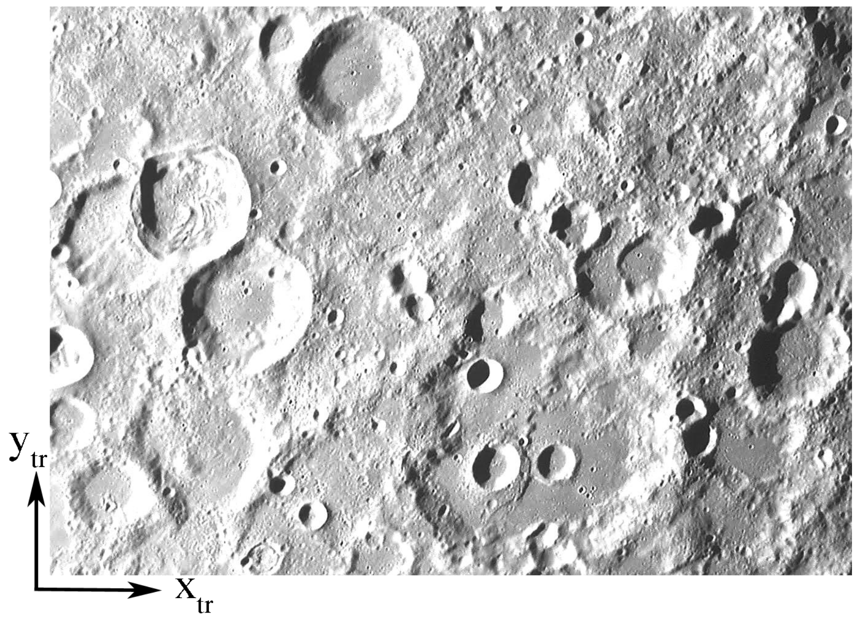


Figure 7.7: Crater Field Overexposed

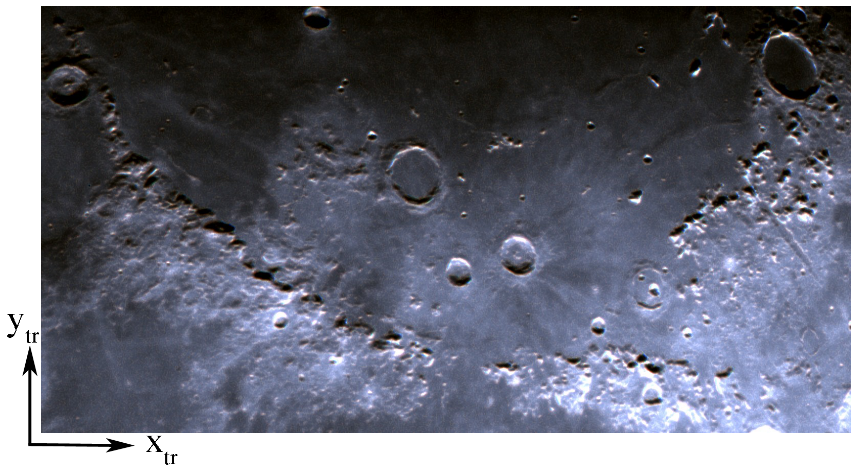


Figure 7.8: Lunar Plains

Figures 7.6 and 7.7 depict the same area of the Moon, ideally with two different brightness settings in order to test the algorithm robustness to various light conditions. The second one presents an higher contrast due to an artificial modification of the projected image fed to the projector. The third sample, shown in Figure 7.8, represents a different area of the Moon and has been selected to assess algorithm performance with different terrain.

The terrain reference frames orientation is shown in all three figures. In accord with these axes it has been decided to define pitching when the camera rotates around y_{tr} axis, yawing when rotating around x_{tr} and rolling when rotating around z_{tr}, not shown in the picture but placed to form a right-hand tern.

7.1.5 Algorithm and acquisition

Data acquisition from the camera-IMU system has been handled by a single PC. The parallel C++ acquisition software has been coded to work on that specific PC. PA10 was controlled by another computer with an already existing C program running on ARTLinux: coupling robotic arm handling and camera-IMU acquisition on the same device was not possible at the time. The computer dedicated to manipulator control handled also data acquisition from the PA10 internal encoders, providing the position and attitude reference used in the tests.

After the acquisition the images from the stream are cropped and and IMU data are pre-processed. Subsequently the two sensors are synchronized and matched with the PA10 data. These data are finally elaborated by Dokuganryuu on the acquisition PC. Direct computation during a test was not possible at the time. The navigation algorithm has been developed on Matlab® using the computer vision toolbox for feature extraction and matching to reduce the development time; the LM algorithm is custom made.

7.2 Assessing the weakness of the full state reconstruction

The final version of Dokuganryuu use gyro measures to determine rotation d.o.f. and images to get the translation d.o.f. The reasoning behind this choice is a possible lack of robustness when the algorithm has to distinguish planar translation from out of plane rotations. This can be verified using the images from two sequences: Horizontal flight and Horizontal flight with pitch.

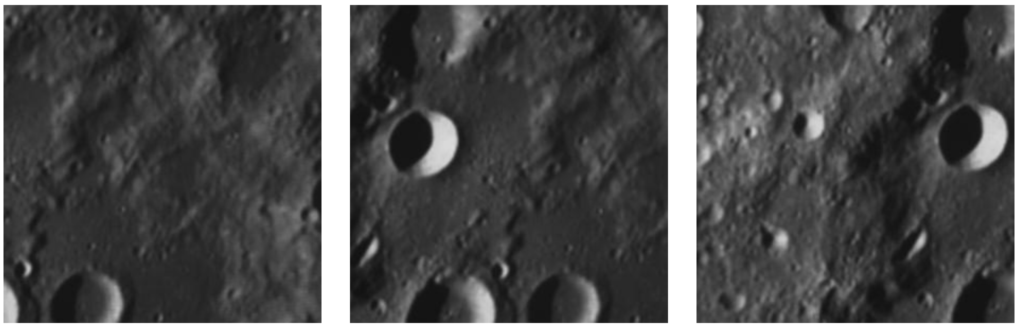

(a) Beginning (b) Half way (c) End

Figure 7.9: Horizontal flight: pure leftward translation

Figure 7.9 present a sequence where the camera is parallel to the ground and it is moving leftward. The motion can be intuitively perceived seeing the craters at the bottom of the first image moving rightward in the following two images. If the camera is considered still, the ground moves rightward, if the ground is still the camera is moving leftward. Like a human being the CV algorithm can reconstruct such behavior. In case of rightward motion the terrain seems to move leftward in a specular case to the one in the picture: looking at the picture from the last up to the first permits to see the specular motion as well.

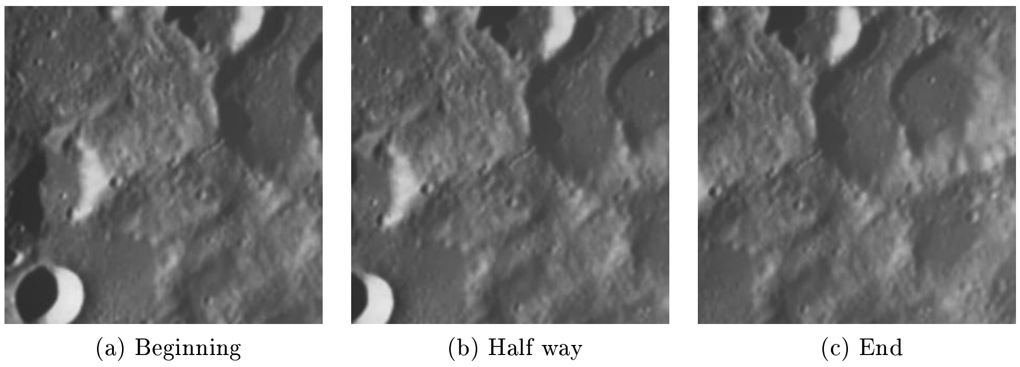


Figure 7.10: Horizontal flight with pitch Leftward translation and rightward rotation

Figure 7.10 images come from a similar sequence where the camera is moving leftward and tilting rightward. The translation direction and magnitude for the whole sequence is the same as the previous, however the camera is subjected to out of plane rotation. Looking at the terrain it seems to move leftward, suggesting a rightward camera motion. The distortion caused by the rotation is not recoverable from one image to the other leading to the drawback of image stream reconstruction that was speculated before.

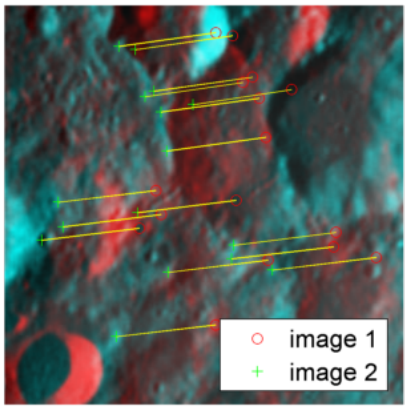


Figure 7.11: Feature matched in Horizontal flight with pitch

Figure 7.11 shows that even when the motion is not very small it is very difficult to find a distortion that might be caused by rotation. The full state reconstruction is not a viable option, Dokuganryuu final configuration might be able to solve this issue through the use of gyroscopes.

7.3 Tests

In this section, obtained experimental results are presented. Test cases have been selected in order to test the most complete as possible combinations of fundamental translations and rotations. In each case, the test is repeated with the same path for each available landing surface. Nominal motion parameters for the experimental sequences are listed in Table 7.4. Velocities are given in terrain reference frame and all simulations lasts 15 seconds.

All the images shown in the section come from real data recorded by the camera during tests. Perspective distortion, visible in some graphs, has been applied with a bi-dimensional model.

Position and attitude error are measured by confronting Dokuganryuu reconstruction with position and attitude computed with the PA10 joint angle measurements. The joints angles positions are used to reconstruct the tip trajectory for their precision, leading to the assumption that they represent the «true motion» of the camera-IMU ensemble. The precision of the reference is deemed to be in the order of the centimeter, since it involves the tip motion while the camera reconstruct its own motion: misplacement of the references might results in apparent error. The initial conditions for Dokuganryuu are taken from this reference and the altitude was measured each test run, since the projector positioning may have varied from case to case.

Table 7.4: Motion test data

Sequence	**Translation Velocity**		**Rotational Velocity**		
	z_{tr} [cm/s]	x_{tr} [cm/s]	x_{tr} [°/s]	y_{tr} [°/s]	z_{tr} [°/s]
Vertical descent	-4	0	0	0	0
Vertical descent with roll	-4	0	0	0	2.5
Horizontal flight	0	-3	0	0	0
Horizontal flight with pitch	0	-3	0	-1	0
Horizontal flight with yaw	0	-3	-0.5	0	0
Diagonal flight	-4	1	0	0	0
Diagonal flight with pitch	-4	1	0	0.5	0

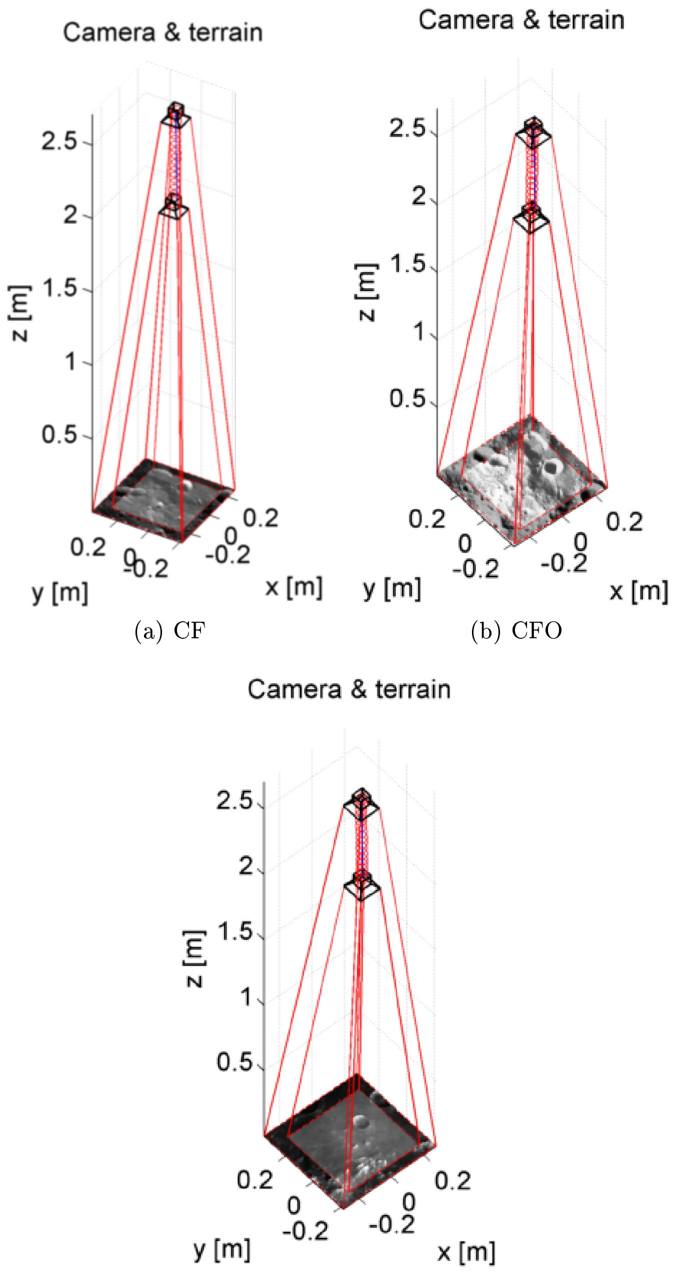
The number of feature processed by Dokuganryuu has not been limited in order to show how their number may affect the precision of the reconstruction. Limiting the number of feature to be processed reduces the computation time, however this important aspects has not been included in the experimental validation of the algorithm due to time and budget constraints.

Another useful insight about the algorithm performance is given by the comparison between the maximum and mean error in the 3D matching with the apparent pixel dimension. The pixel dimension has been obtained through camera intrinsic parameters (**K**) and the local distance of the optics from the screen. The confrontation permits to assess the matching between subsequent images through the LM and its model. Should be noticed that it is an index of LM performance since it is not directly dependent from the motion model coded.

In some cases it has been investigated the precision of the vision system by the introduction of «exact» measures. Rotations and altitude information have been computed using the PA10 measures, i.e. the reference trajectory. Such measures permits to see how the system would have performed in case of better gyroscopic measures or with the addition of an altimeter.

Difference in brightness in cumulative plots are added artificially, with the sole purpose to highlight subsequent image boundaries for abetter readability. A good overlapping between images is itself a representative index of the effectiveness of the proposed method.

7.3.1 Vertical descent



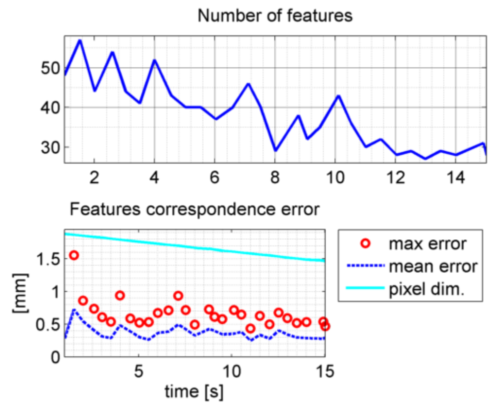
(a) CF

(b) CFO

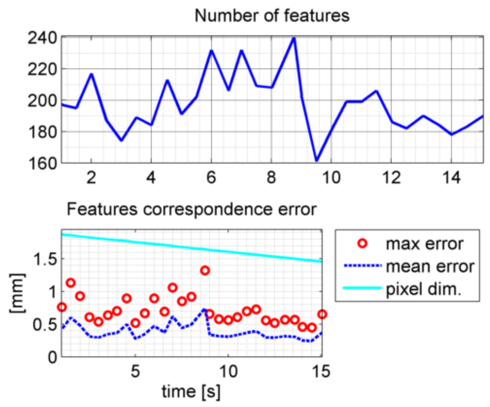
(c) LP

Figure 7.12: Vertical descent - Reconstructed trajectory and terrain

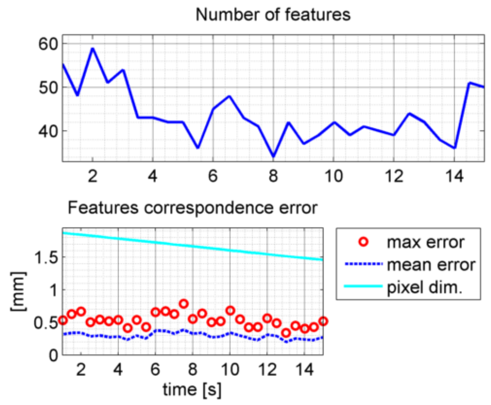
Figure 7.12 presents three simple sequences where the camera is moving on a straight line towards the screen, simulating a vertical descent towards the landing site.



(a) CF



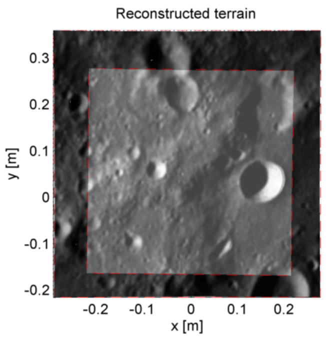
(b) CFO



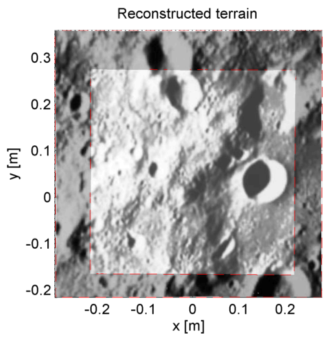
(c) LP

Figure 7.13: Vertical descent - Features and error comparison

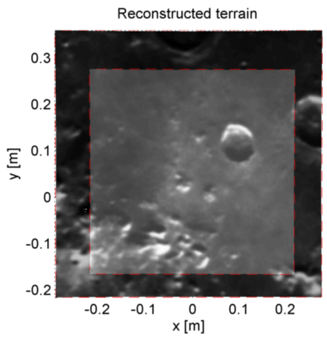
Figure 7.13 shows that the maximum error in feature matching is always smaller than the dimensions of the projected pixel dimension. This implies that the reconstruction of the image stream is smooth: consecutive images are superimposed perfectly. The number of feature shown in Figure 7.13 are affected by the image contrast, in fact the second sequence images produce more features than the other two with lower contrast. This implies that a pre-processing of the images can be used to increase or decrease the number of feature to be tracked. Moreover, the number of feature does not seem to affect the determination error, hence a proper reduction in the number of feature used in the algorithm should not reduce performance and should reduce the computation time.



(a) CF



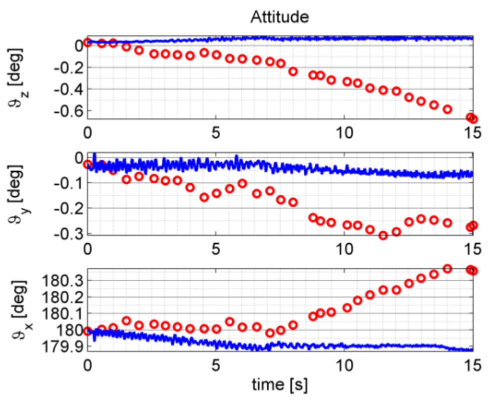
(b) CFO



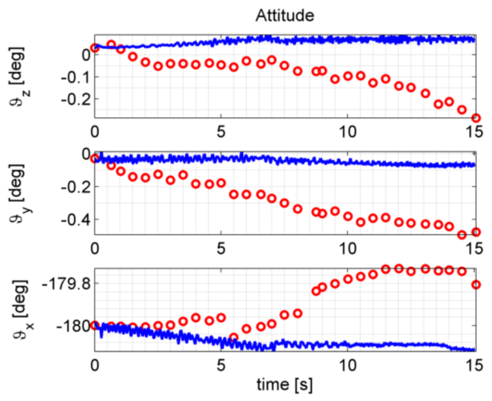
(c) LP

Figure 7.14: Vertical descent - Mapped terrain

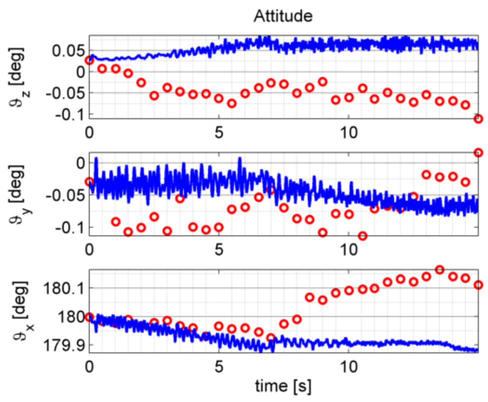
The projection on the ground of reconstructed images is reported in Figure 7.14. for a better readability, only the first and the last processed images are shown. In all the three scenarios a perfect matching is achieved.



(a) CF



(b) CFO

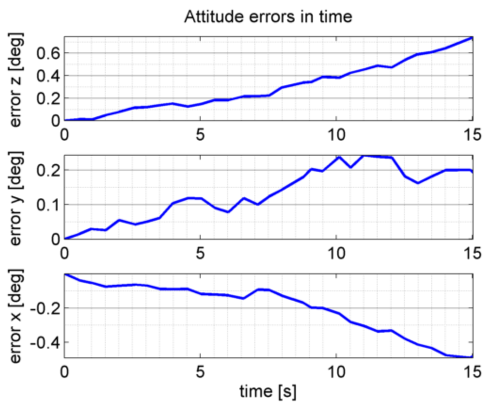


(c) LP

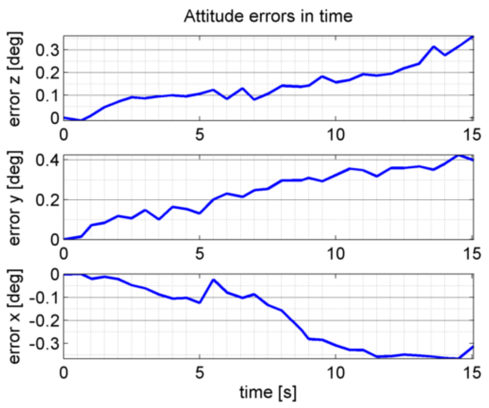
Figure 7.15: Vertical descent - Attitude (gyroscope)
The blue lines represent the PA10 reference while the red circles represent Dokuganryuu reconstruction.

Figure 7.15 reports a comparison between the attitude reconstructed by IMU and

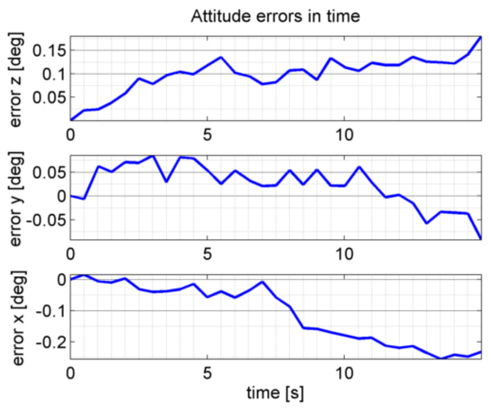
and reference. It is noticeable that some images are not equally spaced in time (this phenomenon can be observed also in other test results). This can be caused by the computer gathering data time code writing or directly by sensor delay. A proper management on a real embedded system should solve this issue.



(a) CF



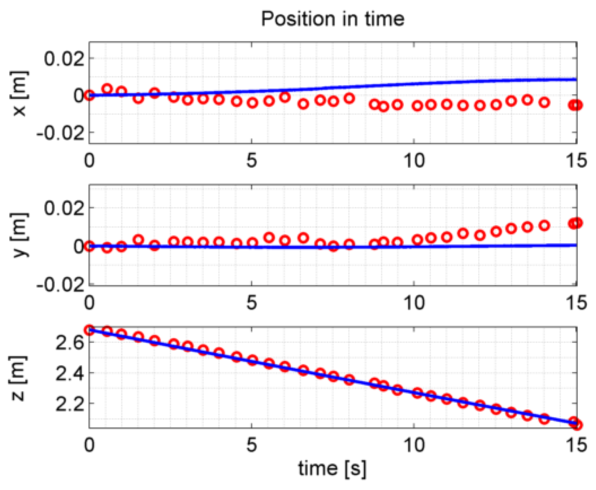
(b) CFO



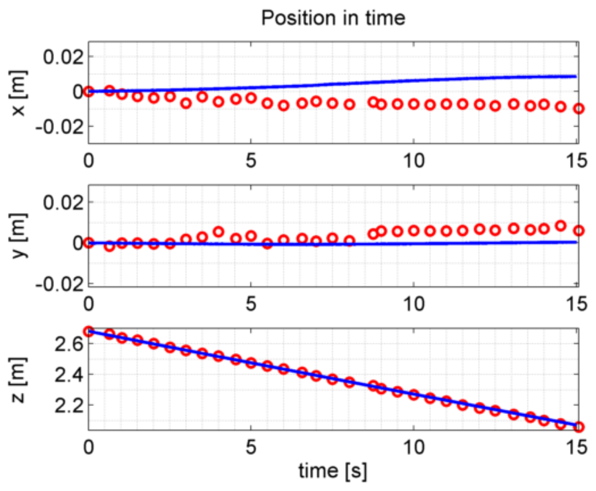
(c) LP

Figure 7.16: Vertical descent - Attitude Error (gyroscope)

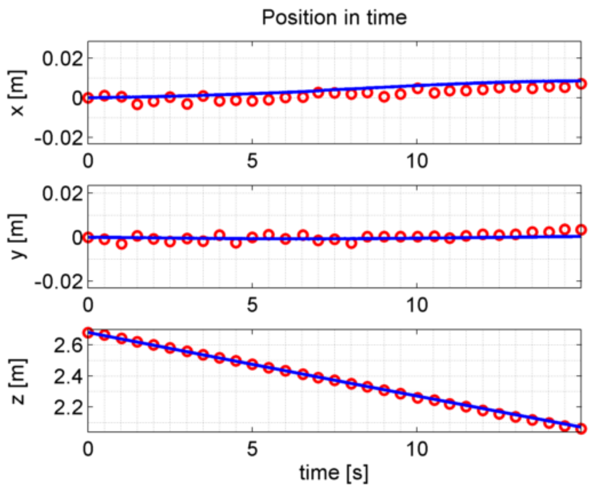
Gyroscope integrated signals drift as expected (Figure 7.16). With a better data filtering or better hardware it is possible to obtain more precise attitude determination, yet it has been decided to keep noisy measurement to manifest and analyze drifting effects. The drift in rotation determination affects indirectly also the position estimation, as seen in Figure 7.17 and 7.18.



(a) CF



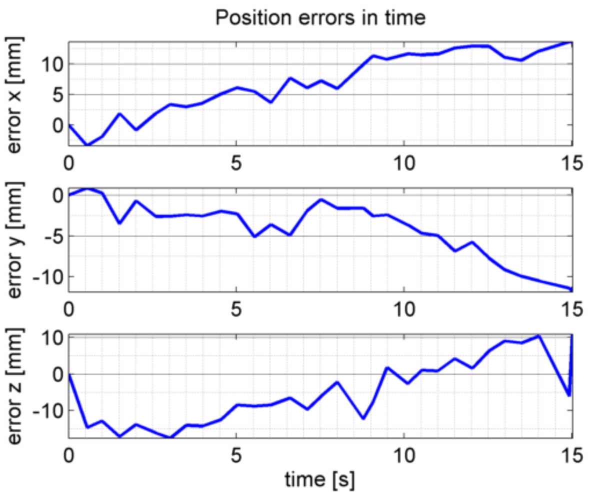
(b) CFO



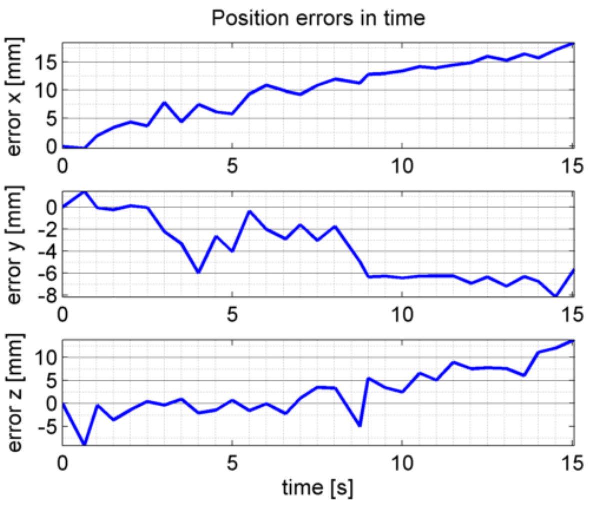
(c) LP

Figure 7.17: Vertical descent - Position

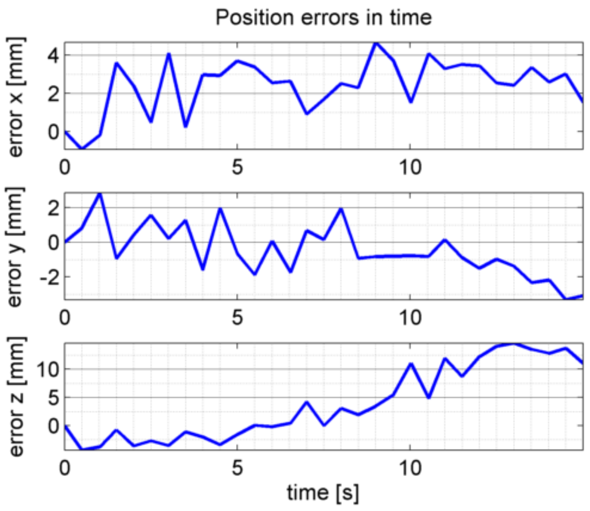
The blue lines represent the PA10 reference while the red circles represent Dokuganryuu reconstruction.



(a) CF



(b) CFO

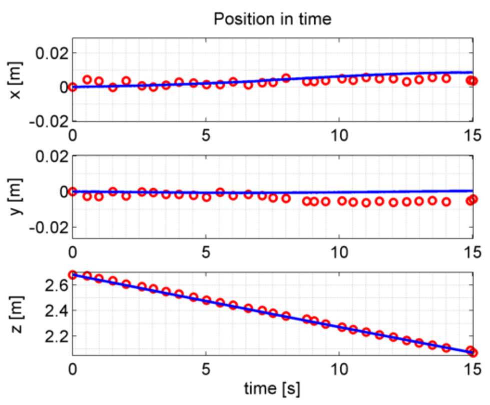


(c) LP

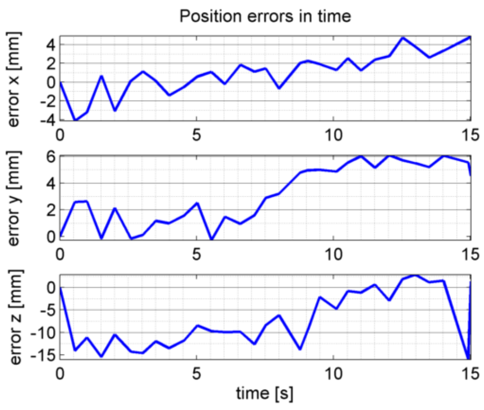
Figure 7.18: Vertical descent - Position Error

Anyway, the macro motion is well recognized by the vision algorithm although gyro drift affects measurement, as seen in Figure 7.17 and 7.18. As expected the position error tends to increase as time goes on. From Figure 7.18c is evident that the drift in position of the LP sequence is lower since the attitude drift is lower than in the first two tests, as clearly shown by Figure 7.16.

7.3.1.1 Vertical descent with perfect attitude



(a) Position



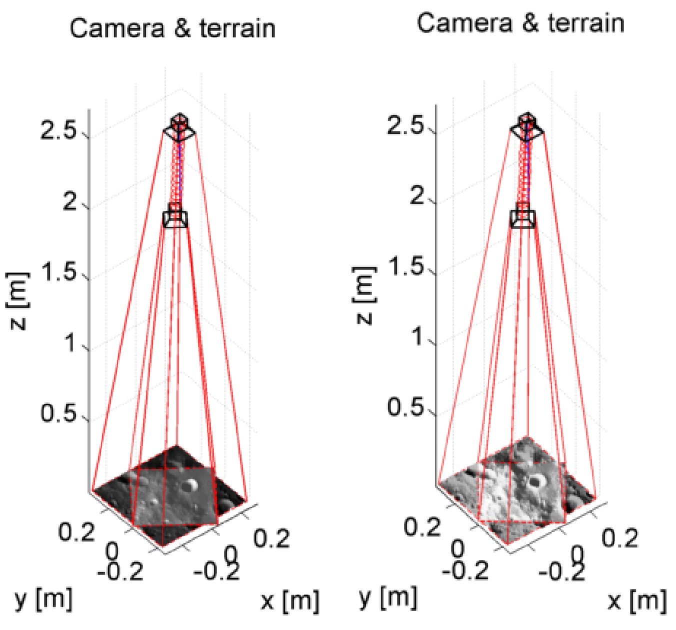
(b) Position error

Figure 7.19: Vertical descent - Perfect attitude (CF sequence)

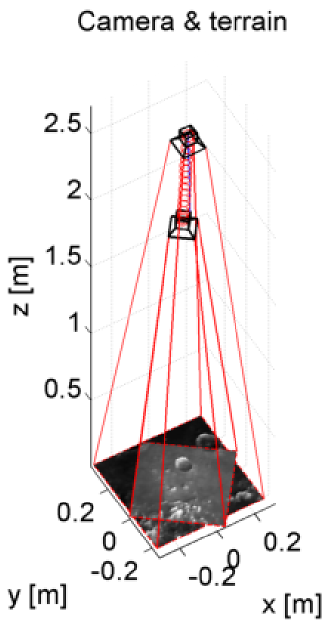
In the first image the blue lines represent the PA10 reference while the red circles represent Dokuganryuu reconstruction.

In order to assess the impact of gyroscopes drift over the reconstructed path, an additional run of the algorithm has been performed considering «true» attitude measures directly from the manipulator. Results reported in Figure 7.19 shows the error in x_{tr} and y_{tr} direction to decrease significantly.

7.3.2 Vertical descent with roll



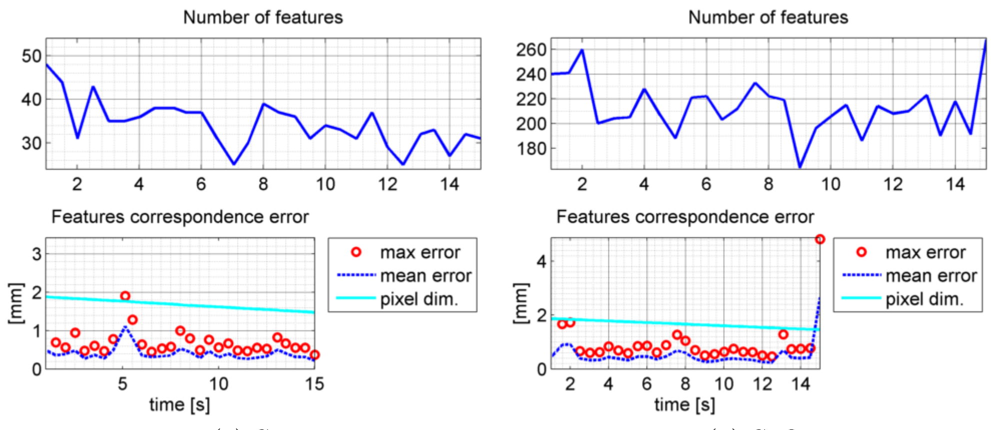
(a) CF (b) CFO



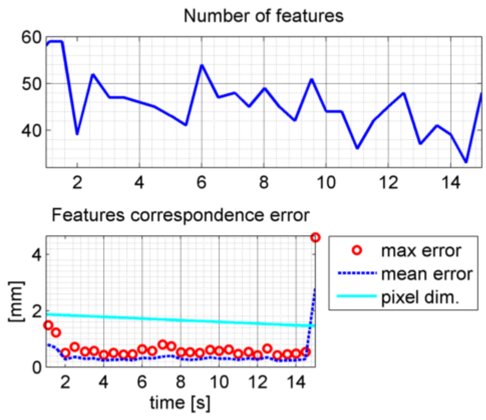
(c) LP

Figure 7.20: Vertical descent with roll - Reconstructed trajectory and terrain

In the sequence depicted in Figure 7.20 the camera performs a screw descent. While descending at constant speed, exactly as the previous sequence, it rotates around z_{tr} axis for 37.5 degrees.



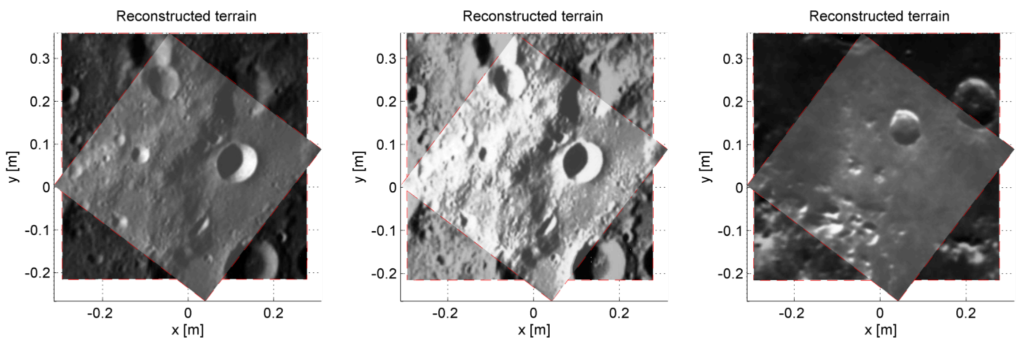
(a) CF (b) CFO



(c) LP

Figure 7.21: Vertical descent with roll - Features and error comparison

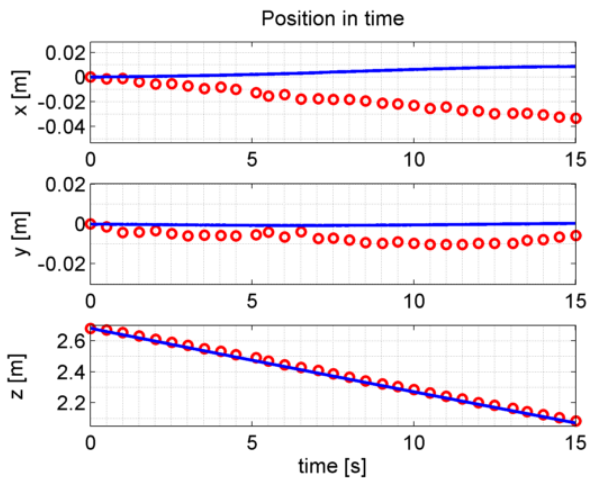
Except some isolated points, the error in 3D reconstruction, visible in Figure 7.21, still remains below the pixel limit. In these sequences the attitude has been reconstructed perfectly from PA10 joints in order to show a possible robustness failure.



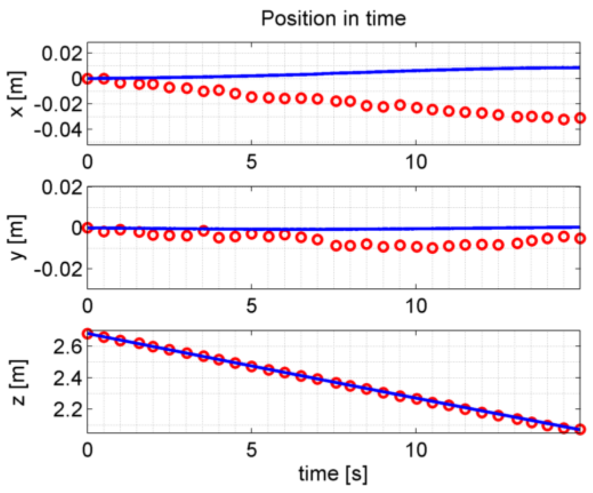
(a) CF (b) CFO (c) LP

Figure 7.22: Vertical descent with roll - Mapped terrain

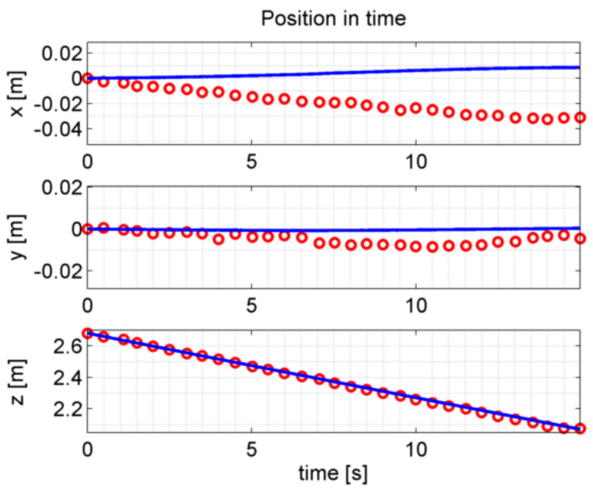
The superimposition of the images still remains perfect, as reported in Figure 7.22.



(a) CF

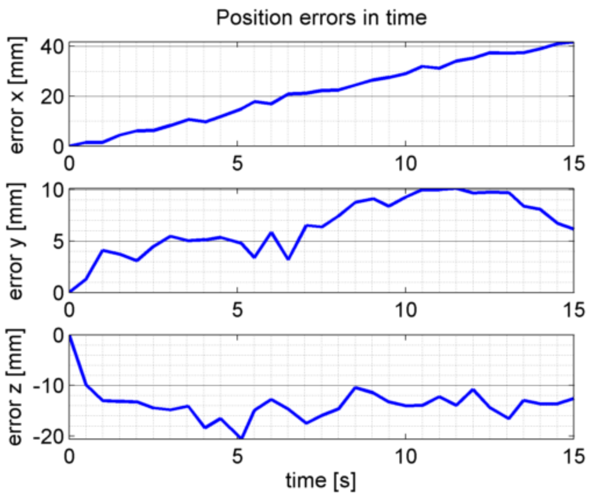


(b) CFO

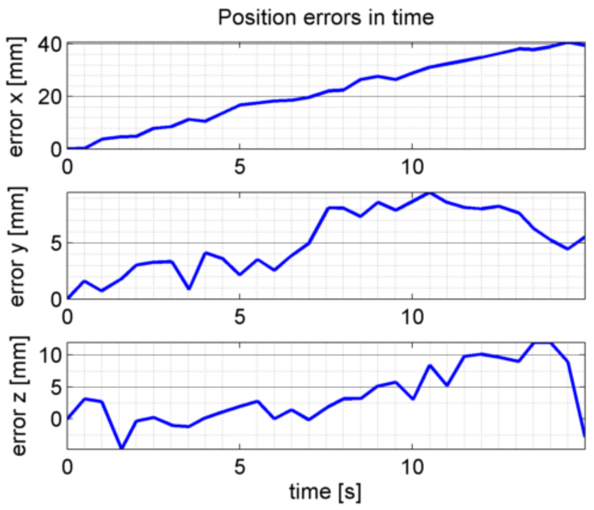


(c) LP

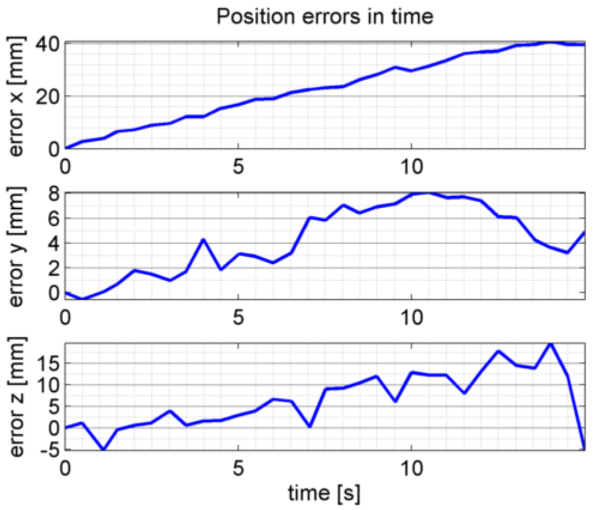
Figure 7.23: Vertical descent with roll - Position
The blue lines represent the PA10 reference while the red circles represent Dokuganryuu reconstruction.



(a) CF



(b) CFO

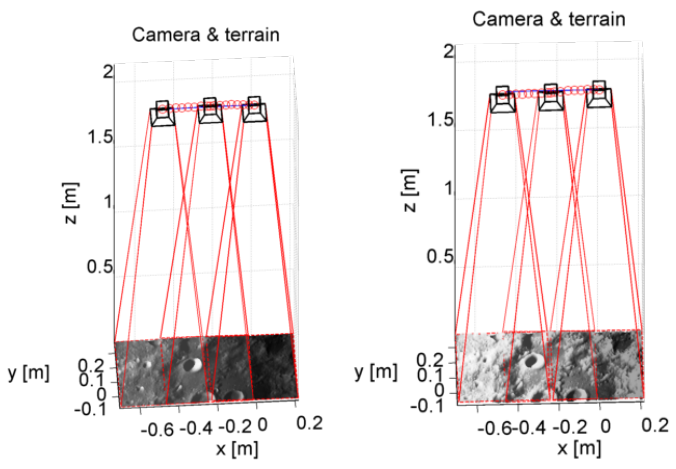


(c) LP

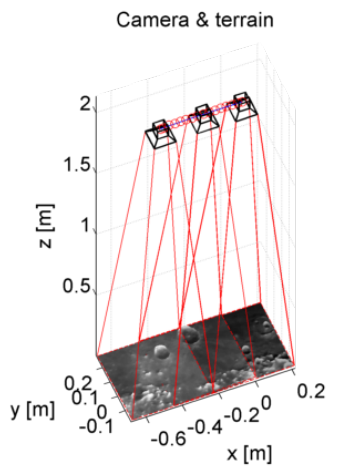
Figure 7.24: Vertical descent with roll - Position Error

Figure 7.23 and 7.24 put in evidence that in all the three tests the camera reconstruction drifts along the x_{tr} axis: the error is not due to attitude reconstruction or LM failure. The best explanation of the phenomena is that the camera is not mounted perfectly on the PA10 tip and the center of rotation is not coincident with the camera optical axis. This misplacement is all but easy to determine and thus suggest to deepen the question for vehicle application. In these experiments it is not possible to evaluate such eccentricity, therefore the reference is not exactly comparable with the reconstruction of the camera, since it is based on the navigation suit motion instead of the PA10 tip motion. Since it is unlikely to have the camera perfectly aligned with the center of mass of the lander it is advisable to estimate the eccentricity. Theoretically it is possible to include the rotation eccentricity in the LM algorithm and determine the center of mass of the lander if it is rotating accordingly.

7.3.3 Horizontal flight



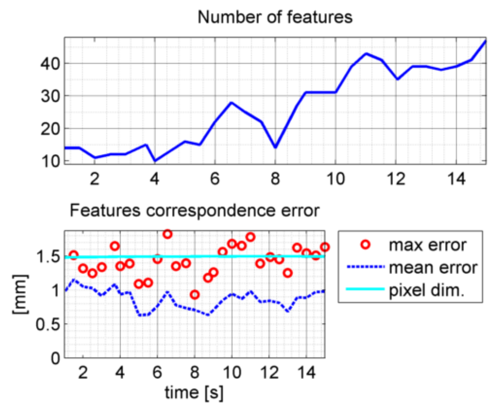
(a) CF (b) CFO



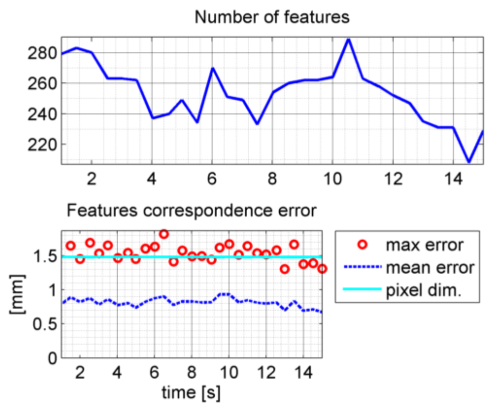
(c) LP

Figure 7.25: Horizontal flight - Reconstructed trajectory and terrain

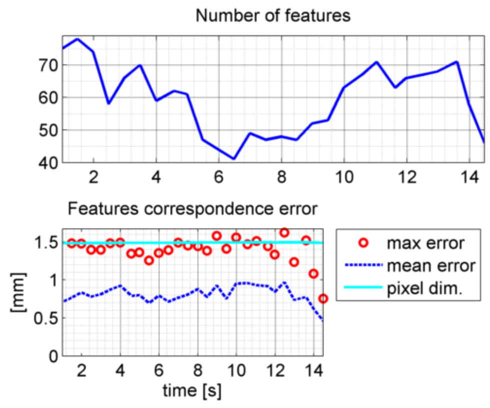
In this sequence the camera moves along a straight line at the constant altitude and a constant speed of $-0.03\,\mathrm{m/s}$ along x_{tr} axis (Figure 7.25).



(a) CF



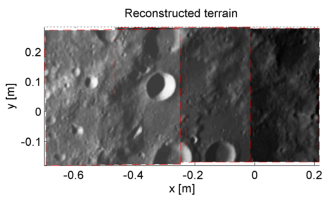
(b) CFO



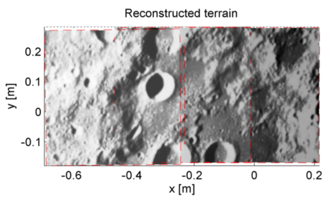
(c) LP

Figure 7.26: Horizontal flight - Features and error comparison

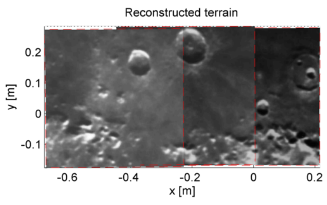
From Figure 7.26 the obtained matching error magnitude is greater than the previous cases of Vertical descent. This is probably due to the larger displacement between features. The maximum error still remains on the same order of magnitude of projected pixels, giving a good reconstruction of the landscape from the stream of images.



(a) CF



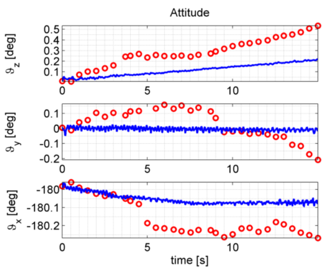
(b) CFO



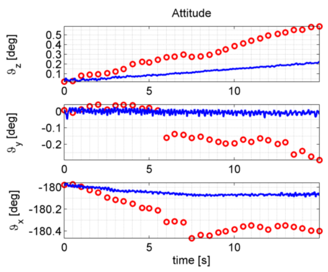
(c) LP

Figure 7.27: Horizontal flight - Mapped terrain

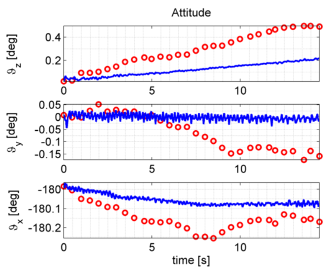
The projected images from Figure 7.27 show once again that the vision system works fine and the estimated position is coherent with the model it has been given.



(a) CF



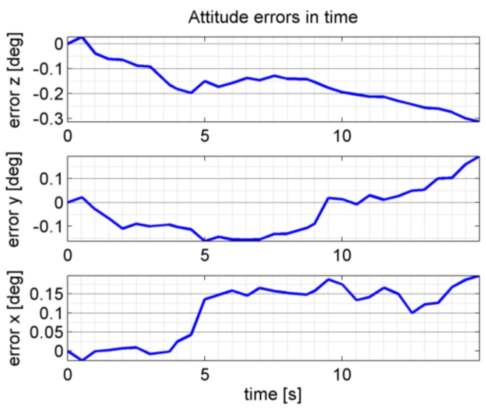
(b) CFO



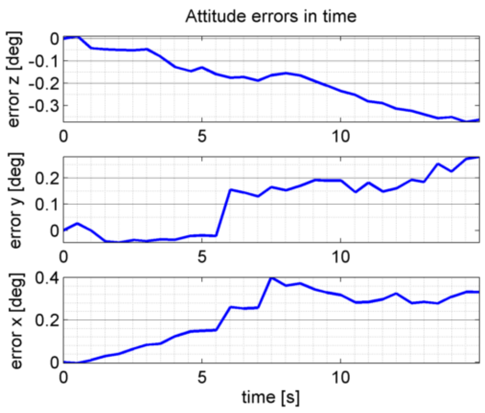
(c) LP

Figure 7.28: Horizontal flight - Attitude (gyroscope)

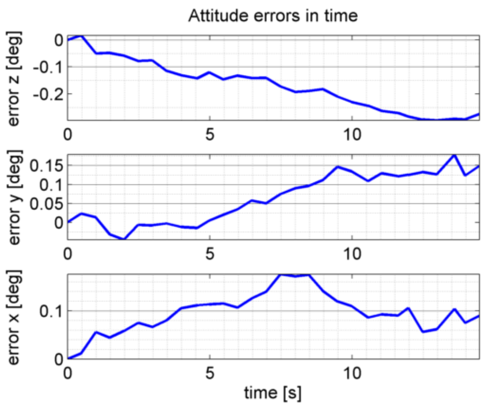
The blue lines represent the PA10 reference while the red circles represent Dokuganryuu reconstruction.



(a) CF



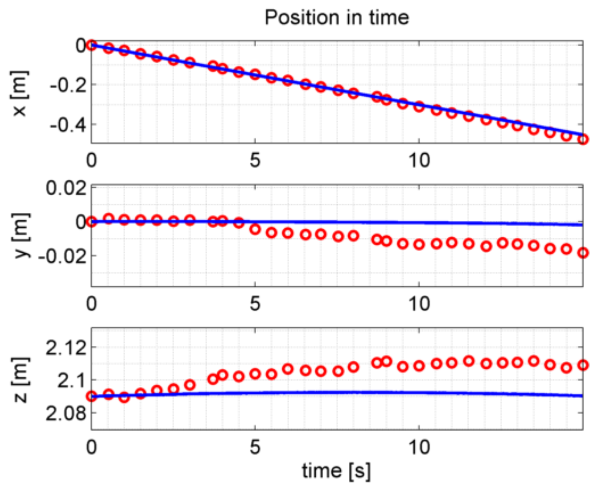
(b) CFO



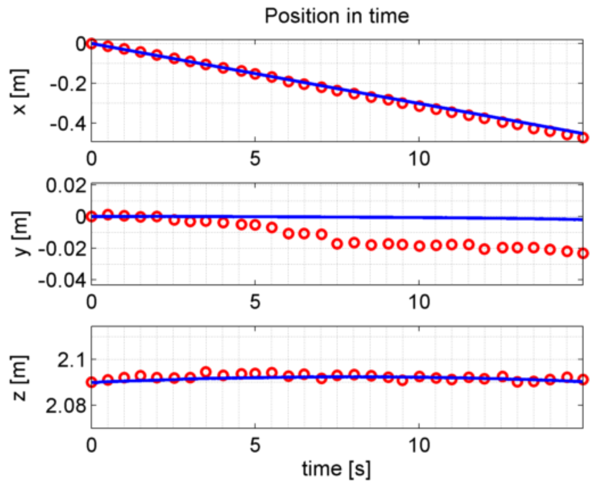
(c) LP

Figure 7.29: Horizontal flight - Attitude Error (gyroscope)

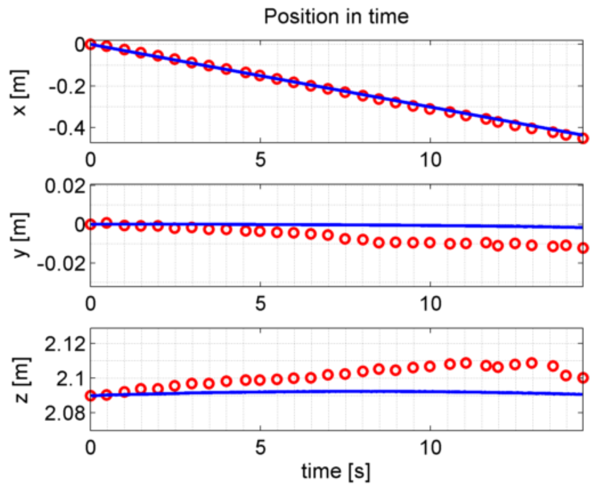
Figures 7.28 and 7.29 shows that the measured attitude error is always lower than 0.5 degrees at the end of the test. Despite of this low value, this error still remains an indirect source of errors also in position determination, as can be seen in Figures 7.30 and 7.31.



(a) CF



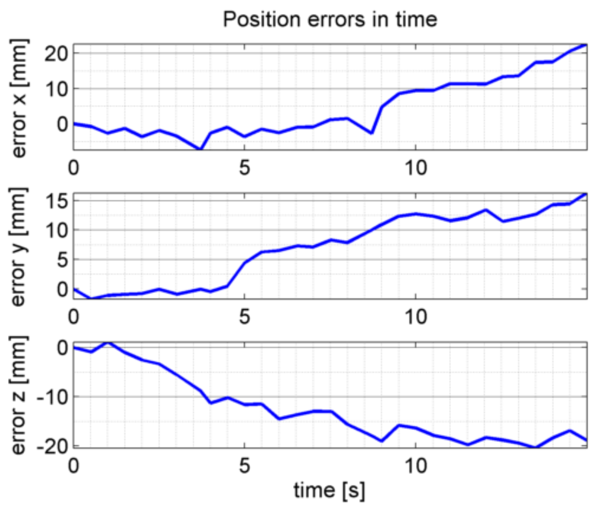
(b) CFO



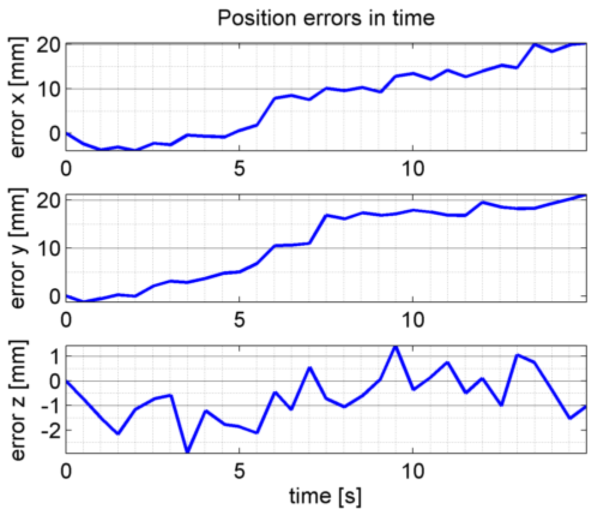
(c) LP

Figure 7.30: Horizontal flight - Position

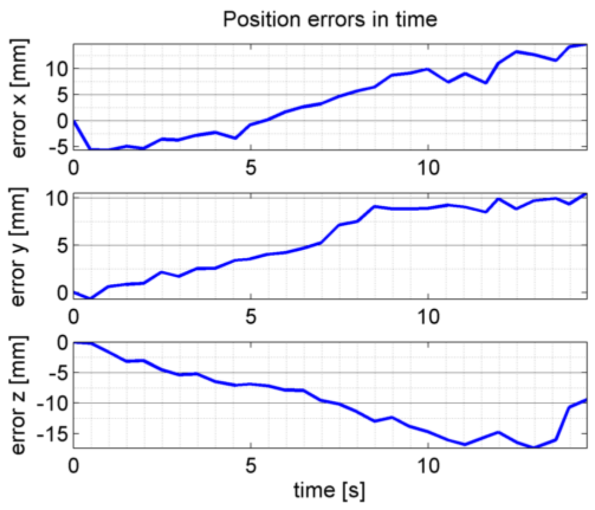
The blue lines represent the PA10 reference while the red circles represent Dokuganryuu reconstruction.



(a) CF



(b) CFO

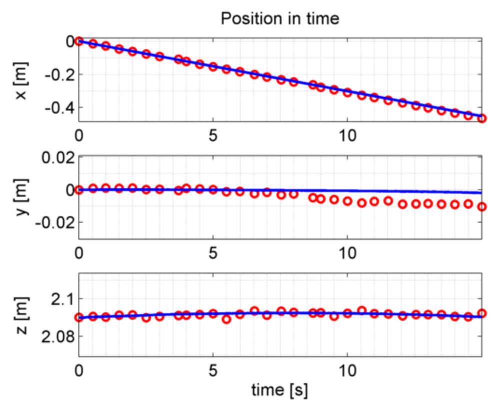


(c) LP

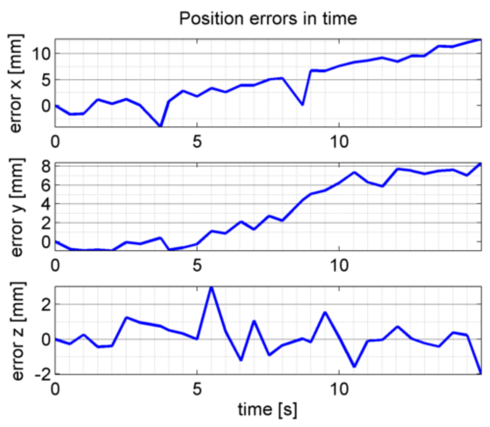
Figure 7.31: Horizontal flight - Position Error

The drifts visible in Figure 7.30 and 7.31 are more significant than in the previous Vertical descent tests. The most relevant component is the drift in altitude, since it provokes more error in the other two directions, being extremely correlated to the scale of the image 3-D points. An error in altitude results in an increased error in the other two directions, therefore reducing this error would increase the robustness of the system.

7.3.3.1 Horizontal flight with perfect attitude and altimeter



(a) Position

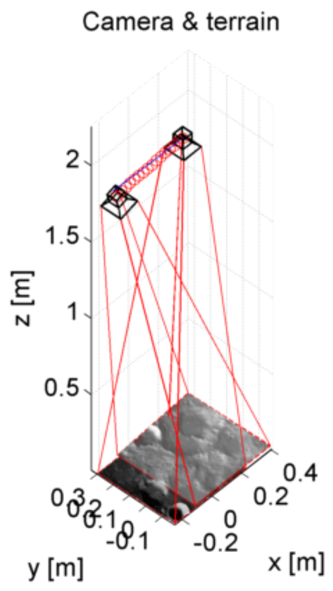


(b) Position Error

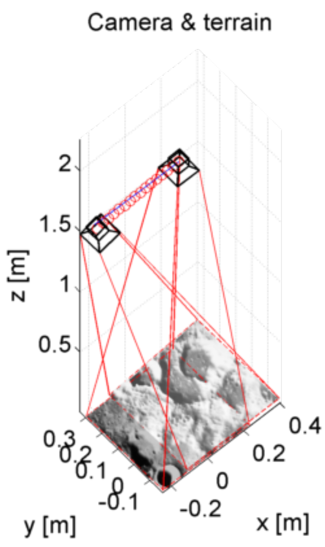
Figure 7.32: Horizontal flight - Perfect attitude and altimeter measure (CF sequence)
In the first image the blue lines represent the PA10 reference while the red circles represent Dokuganryuu reconstruction.

Figure 7.32 shows that if the attitude error is removed and the altitude is measured by a somehow precise instrument different from the camera (radar or laser altimeter for instance) the drift is highly reduced obtaining both precision and robustness.

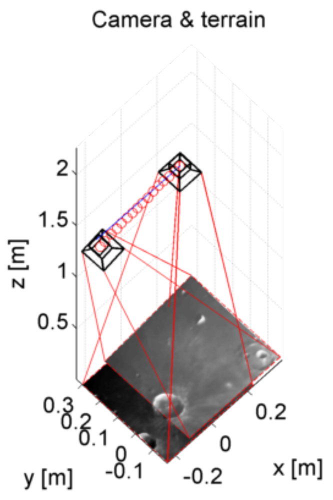
7.3.4 Horizontal flight with pitch



(a) CF



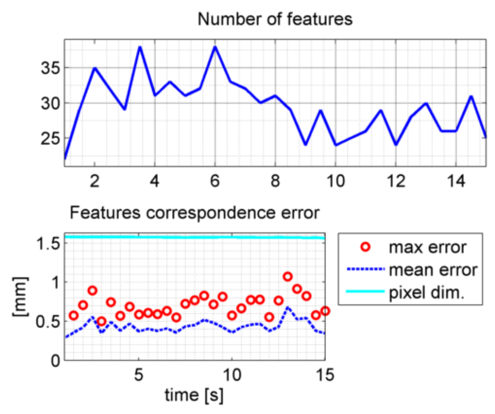
(b) CFO



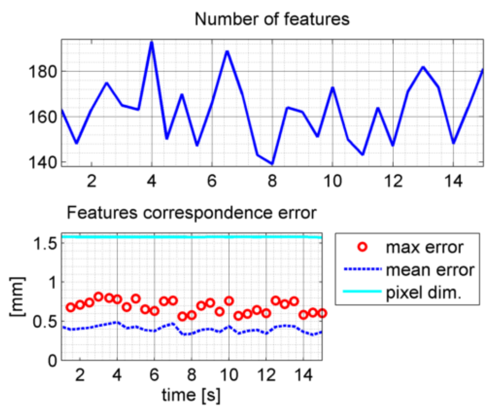
(c) LP

Figure 7.33: Horizontal flight with pitch - Reconstructed trajectory and terrain

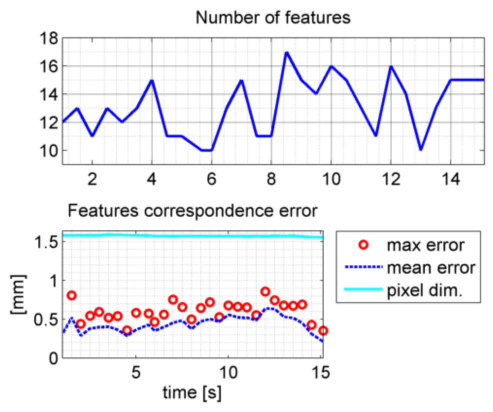
The sequence presented in Figure 7.33 follows the same trajectory of the previous one, but a constant angular rate of $-1°/s$ is imposed around y_{tr} axis resulting in a rightward pitching motion. The horizontal motion is coupled with a rotation that has more effect on the vision than the translation itself, resulting in an apparent backward motion. Although borderline, this sequence helps to address the limits of the system.



(a) CF



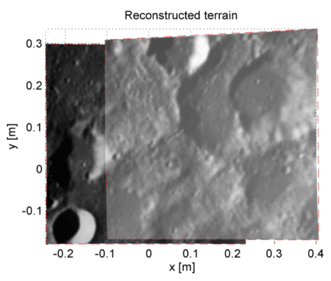
(b) CFO



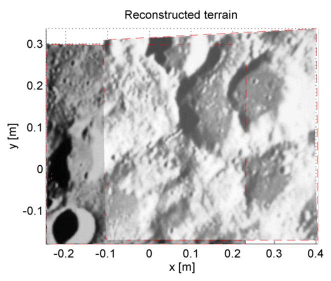
(c) LP

Figure 7.34: Horizontal flight with pitch - Features and error comparison

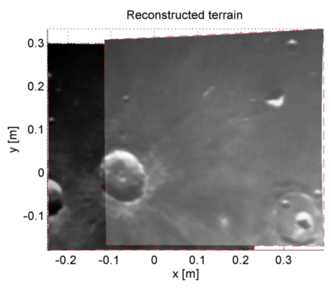
In opposition with the previous simulation, the combined rototranslation points the camera towards the same portion of the surface, with a small displacement between views. This of course influence the error shown in Figure 7.34 that is more aligned with the Vertical descent tests.



(a) CF



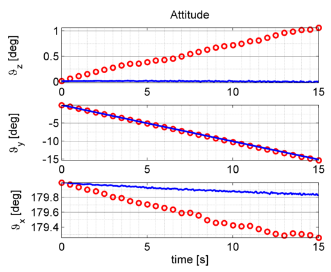
(b) CFO



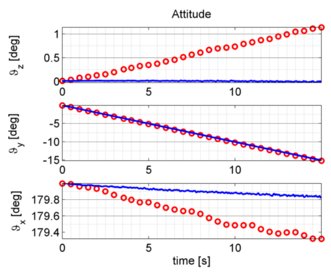
(c) LP

Figure 7.35: Horizontal flight with pitch - Mapped terrain

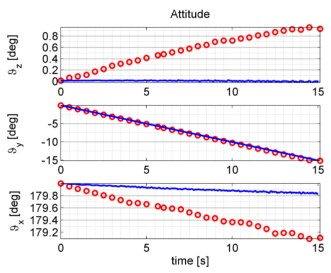
Also in this case the the matching error remains always below the projected pixel size, the images superimpositions in Figure 7.35 are coherent.



(a) CF

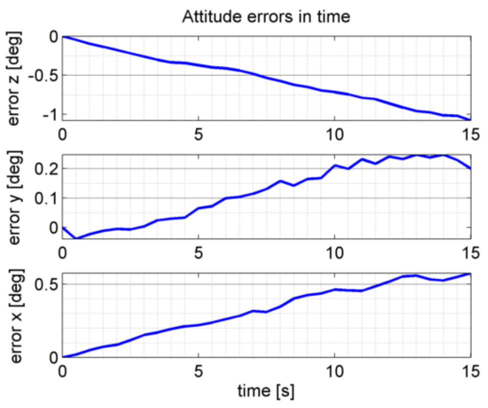


(b) CFO

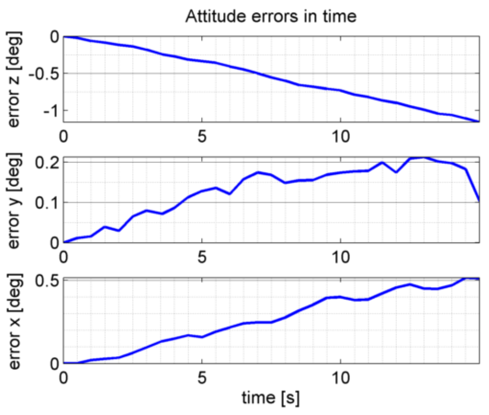


(c) LP

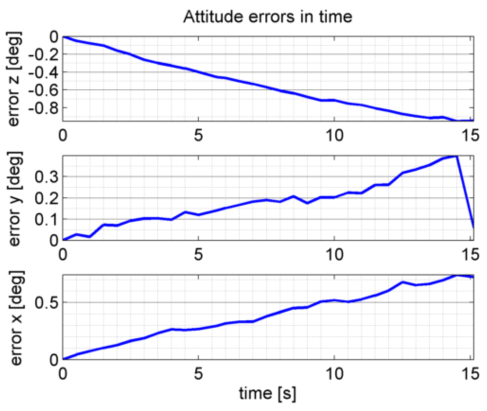
Figure 7.36: Horizontal flight with pitch - Attitude (gyroscope)
The blue lines represent the PA10 reference while the red circles represent Dokuganryuu reconstruction.



(a) CF



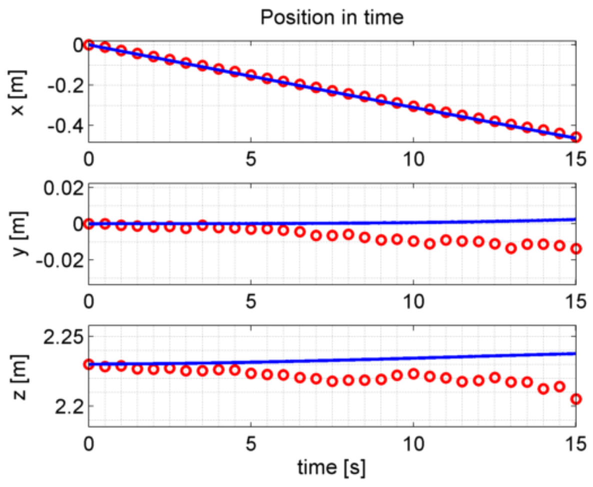
(b) CFO



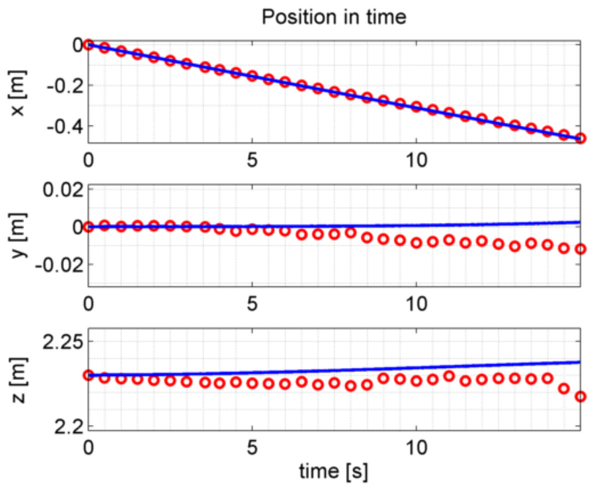
(c) LP

Figure 7.37: Horizontal flight with pitch - Attitude Error (gyroscope)

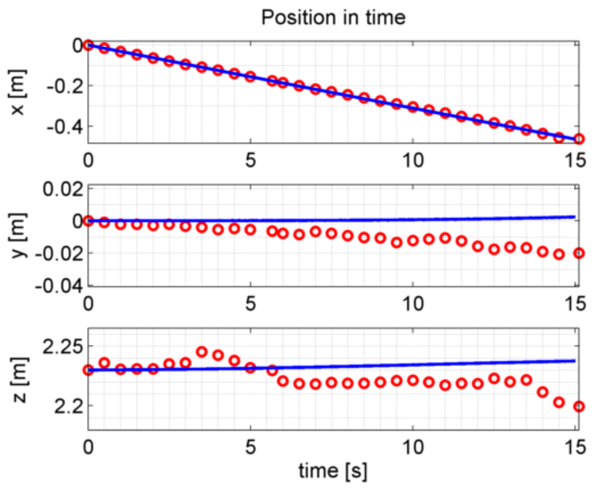
In these particular cases the attitude error drifts of about one full degree (Figure 7.36 and 7.37).



(a) CF

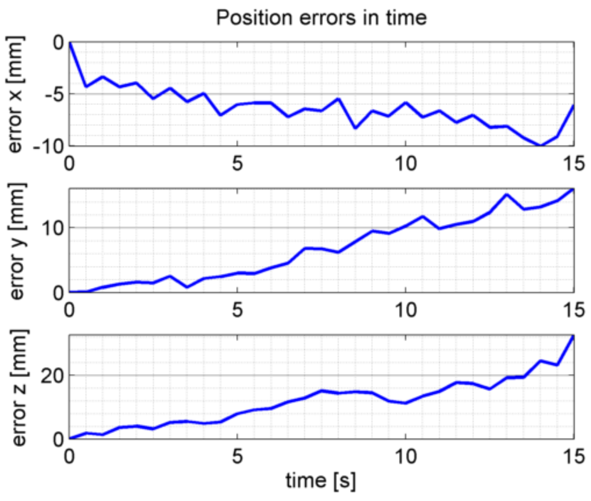


(b) CFO

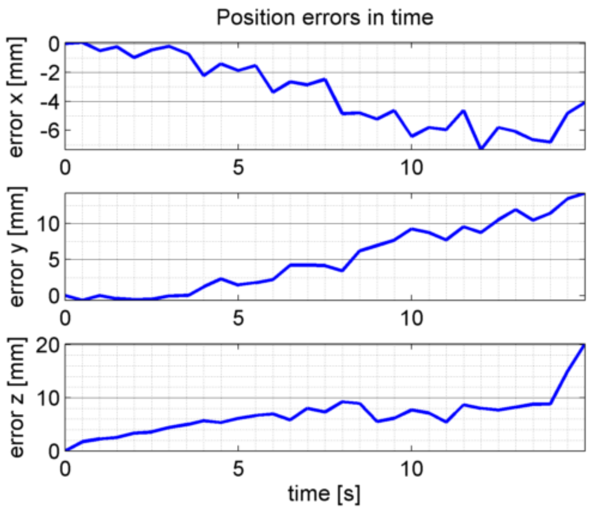


(c) LP

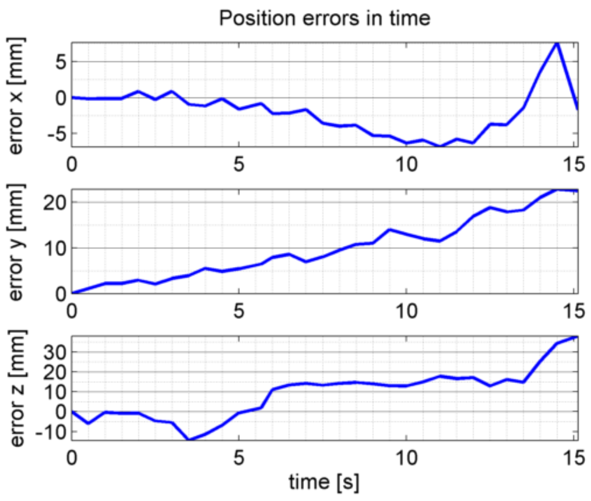
Figure 7.38: Horizontal flight with pitch - Position
The blue lines represent the PA10 reference while the red circles represent Dokuganryuu reconstruction.



(a) CF



(b) CFO

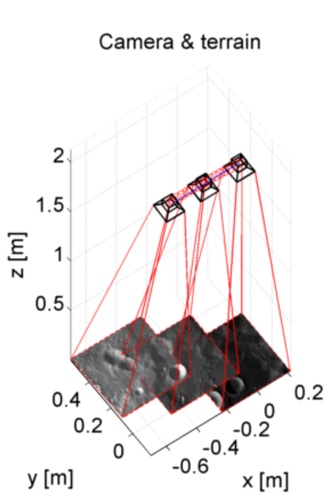


(c) LP

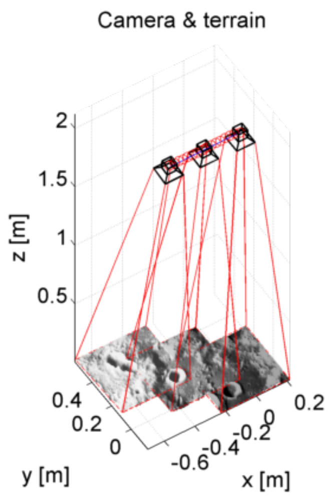
Figure 7.39: Horizontal flight with pitch - Position Error

As could be expected from a vision reconstruction of a borderline sequence the errors and drift are evident (Figure 7.39), although the left macro motion is still captured (Figure 7.38). The use of gyroscopes proves to be necessary, looking at the results. This confirms that the speculation over out of plane rotations was correct and that Dokuganryuu is capable of reconstructing the trajectory even in these cases.

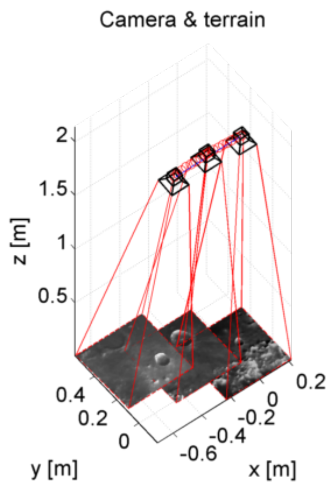
7.3.5 Horizontal flight with yaw



(a) CF



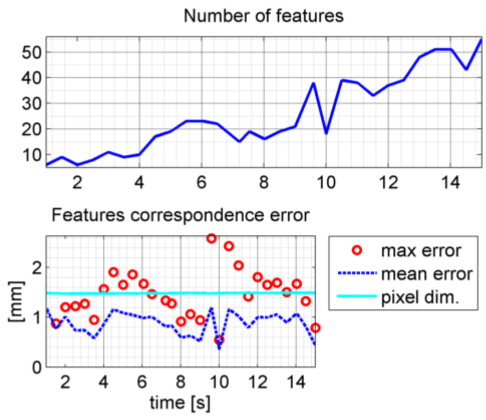
(b) CFO



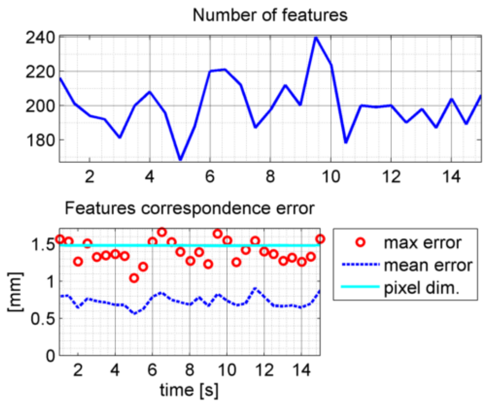
(c) LP

Figure 7.40: Horizontal flight with yaw - Reconstructed trajectory and terrain

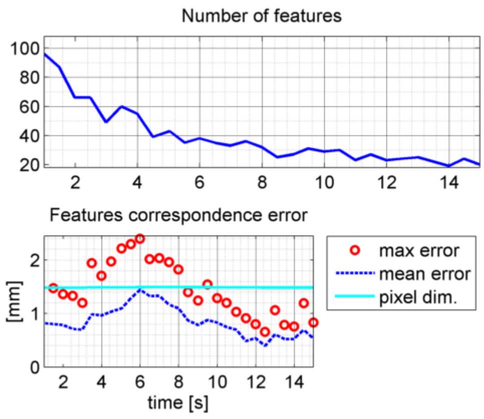
This sequence exploit the same leftward motion as Horizontal flight and Horizontal flight with pitch, but add a constant rotation around the x_{tr} axis, as seen in Figure 7.40. Without the use of gyroscope the computer vision algorithm would have probably reconstructed a translation in both on x_{tr} and y_{tr}.



(a) CF



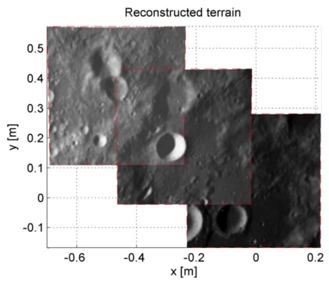
(b) CFO



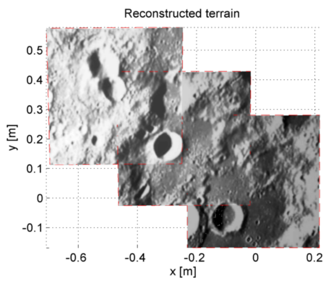
(c) LP

Figure 7.41: Horizontal flight with yaw - Features and error comparison

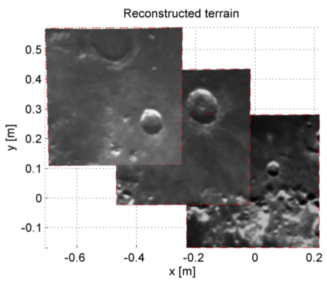
As for the case of Section 7.3.3, the relative large displacement between matched features causes an increase in their displacement error, still under the pixel dimension threshold (7.41).



(a) CF



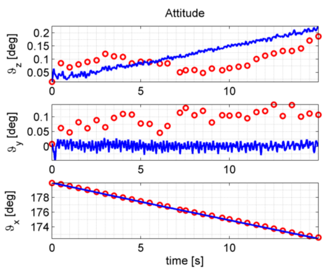
(b) CFO



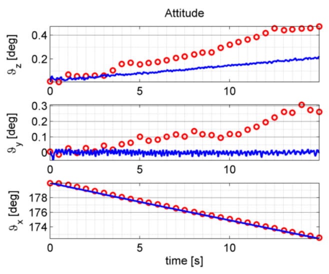
(c) LP

Figure 7.42: Horizontal flight with yaw - Mapped terrain

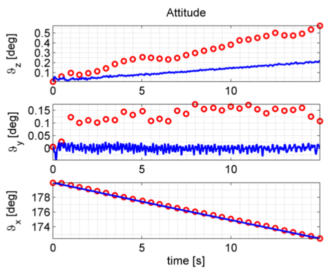
Despite the errors in 3-D feature matching, the projected images in Figure 7.42 are well overlapped.



(a) CF



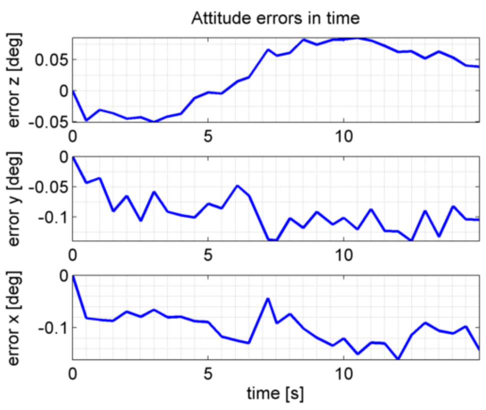
(b) CFO



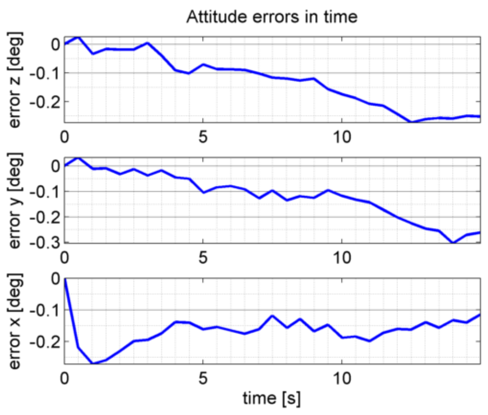
(c) LP

Figure 7.43: Horizontal flight with yaw - Attitude (gyroscope)

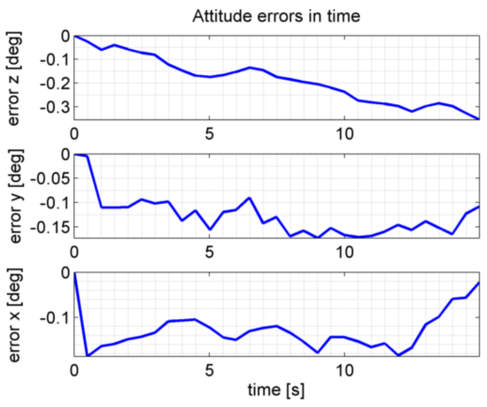
The blue lines represent the PA10 reference while the red circles represent Dokuganryuu reconstruction.



(a) CF



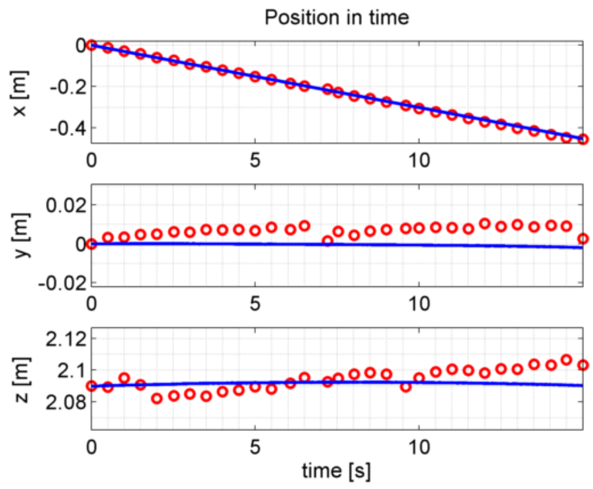
(b) CFO



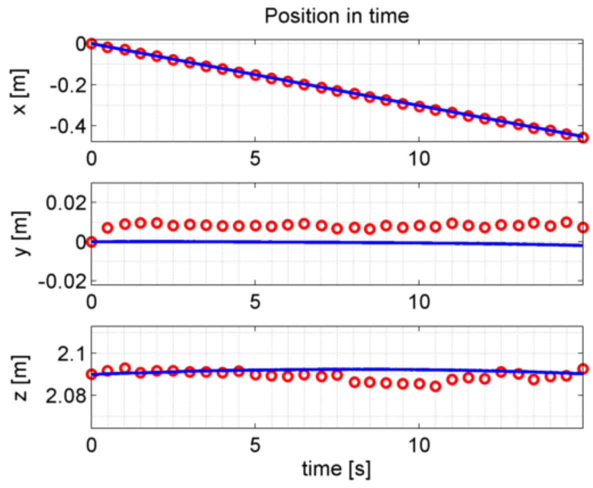
(c) LP

Figure 7.44: Horizontal flight with yaw - Attitude Error (gyroscope)

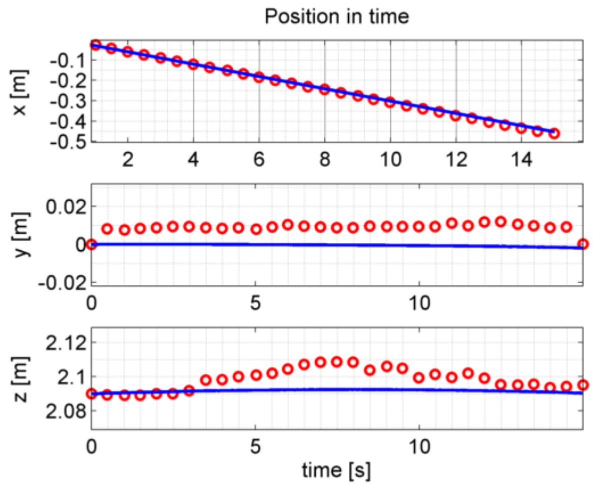
In Figure 7.44 is possible to see that the roll error presents a sudden increase just at the beginning of the maneuver.



(a) CF



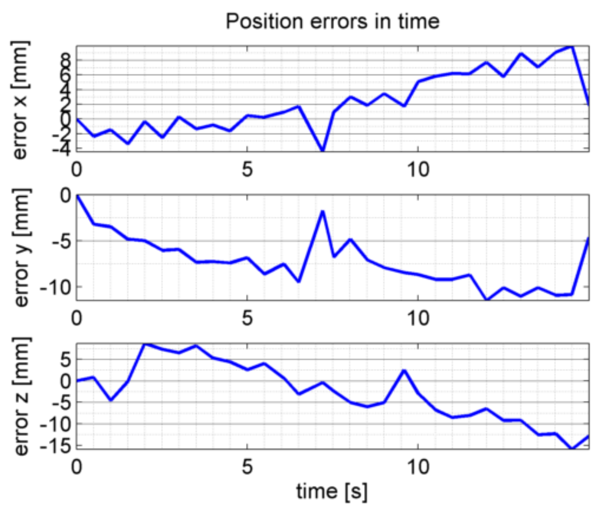
(b) CFO



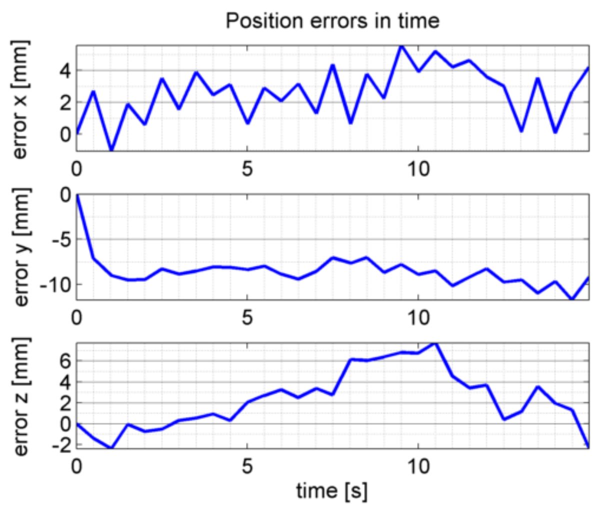
(c) LP

Figure 7.45: Horizontal flight with yaw - Position

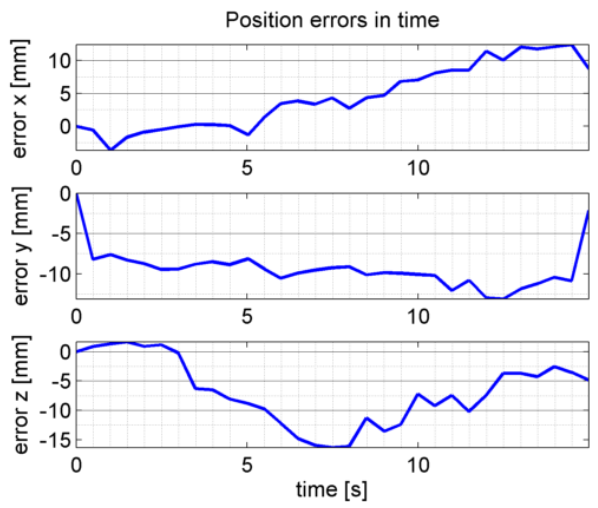
The blue lines represent the PA10 reference while the red circles represent Dokuganryuu reconstruction.



(a) CF



(b) CFO

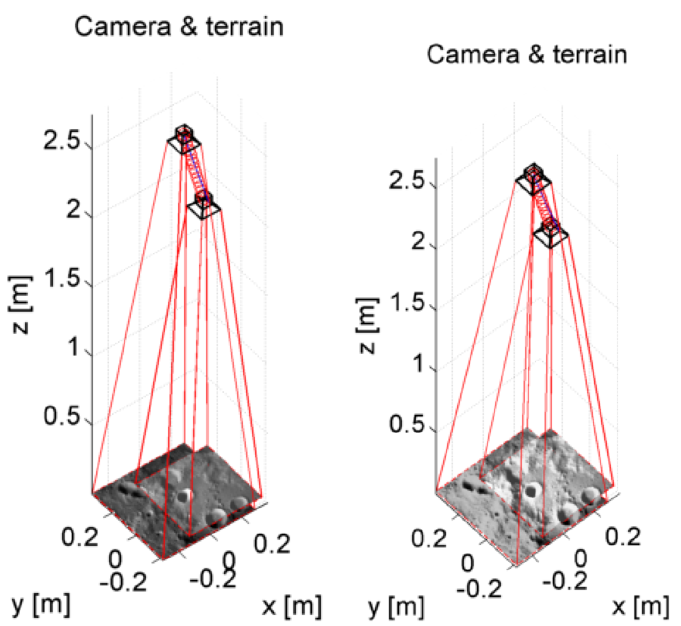


(c) LP

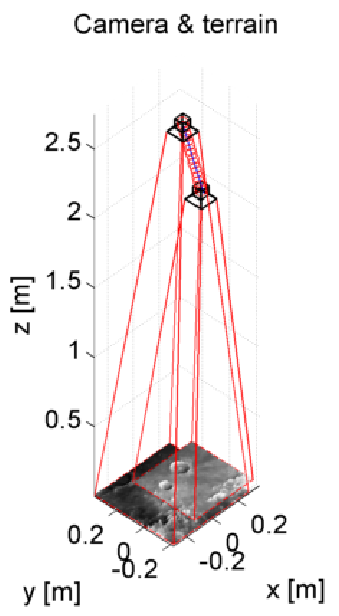
Figure 7.46: Horizontal flight with yaw - Position Error

Figures 7.45 and 7.46 shows an initial error in the reconstruction of y_{tr} direction that produces somehow a constant error. This is a typical example of system drift: an error in reconstruction affects permanently all the subsequent estimations. Unless a proper filter modeled on the lander dynamics is designed or some limits are removed, there is no way that such a system can recover the error. The tracking of features in a longer time window should increase the robustness, but this is possible only if features remain in sight for enough frames. In a landing scenario, this could happen only in the very last phases, when the horizontal speed is low and the trajectory is near vertical.

7.3.6 Diagonal flight



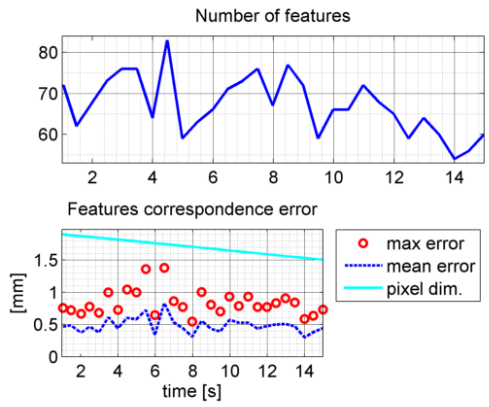
(a) CF (b) CFO



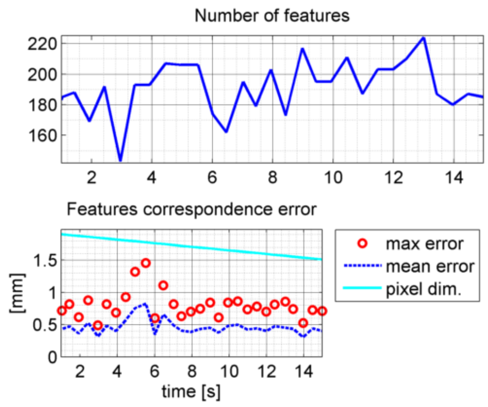
(c) LP

Figure 7.47: Diagonal flight - Reconstructed trajectory and terrain

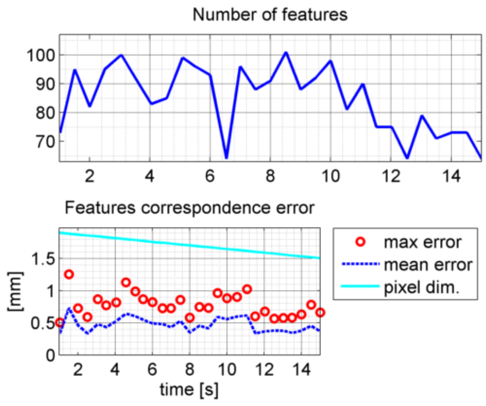
This motion sequence combines pure translation along two directions: x_{tr} and z_{tr}. The camera is mounted like the Vertical descent case and it moves towards the screen at $-0.04\,\mathrm{m/s}$ and partly on the x_{tr} axis with a velocity of $0.01\,\mathrm{m/s}$. The trajectory is shown in Figure 7.47.



(a) CF



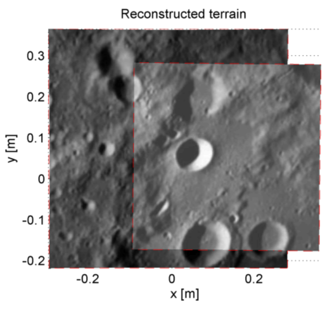
(b) CFO



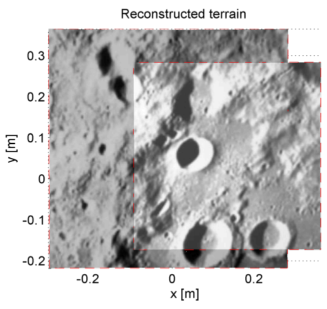
(c) LP

Figure 7.48: Diagonal flight - Features and error comparison

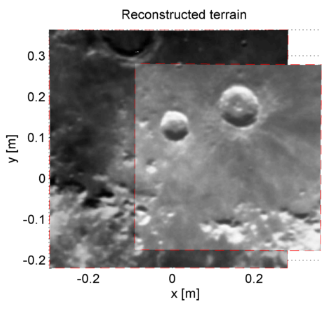
Since the observing camera window is looking more or less at the same portion of land the feature displacement in Figure 7.48 is contained and thus the error is below pixel size.



(a) CF



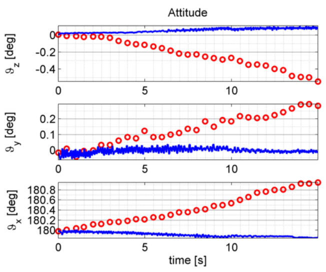
(b) CFO

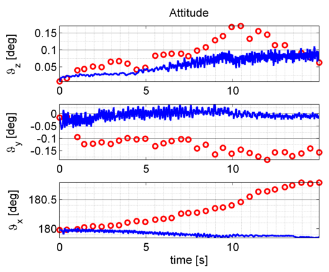


(c) LP

Figure 7.49: Diagonal flight - Mapped terrain

The good images overlapping shown in Figure 7.49 confirms the considerations based on the good matching error of Figure 7.48.


(a) CF


(b) CFO

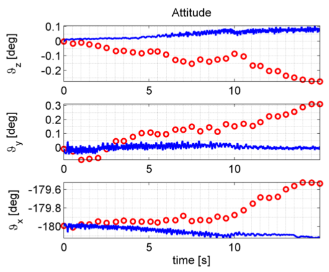
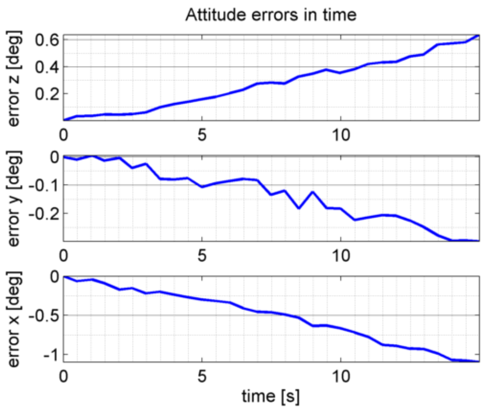
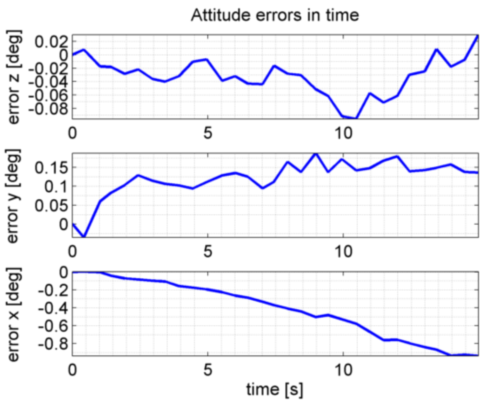

(c) LP

Figure 7.50: Diagonal flight - Attitude (gyroscope)

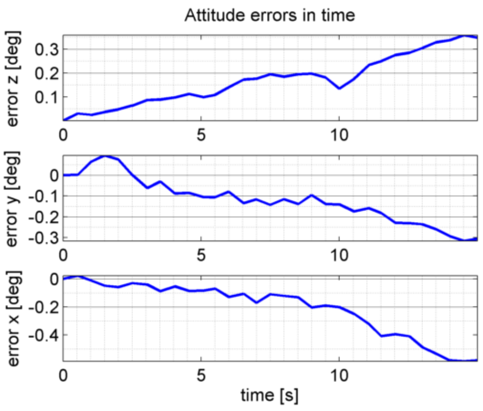
The blue lines represent the PA10 reference while the red circles represent Dokuganryuu reconstruction.



(a) CF



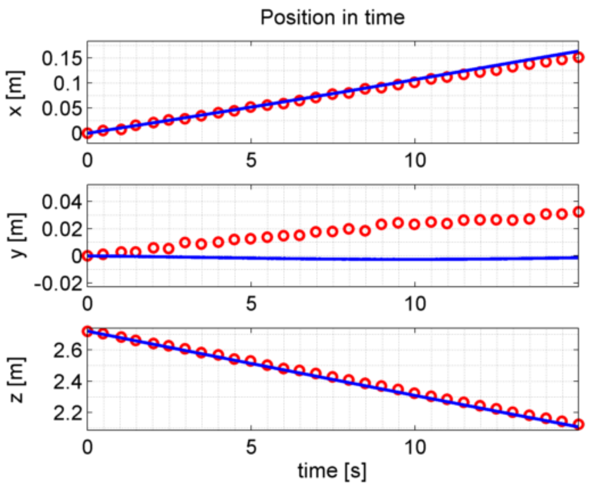
(b) CFO



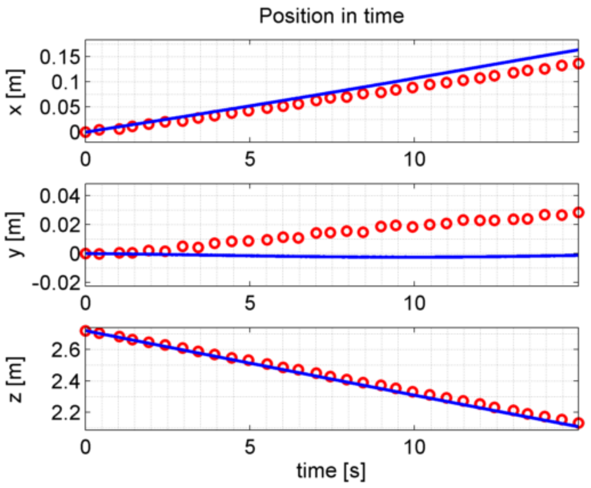
(c) LP

Figure 7.51: Diagonal flight - Attitude Error (gyroscope)

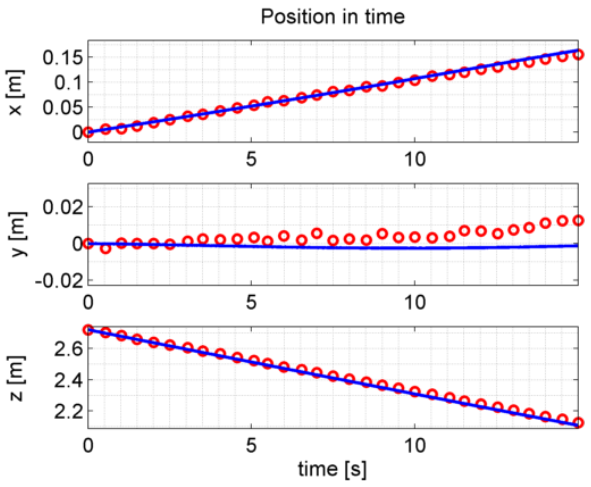
The gyro drift is mostly significant just in one axis, as shown in Figures 7.50 and 7.51.



(a) CF



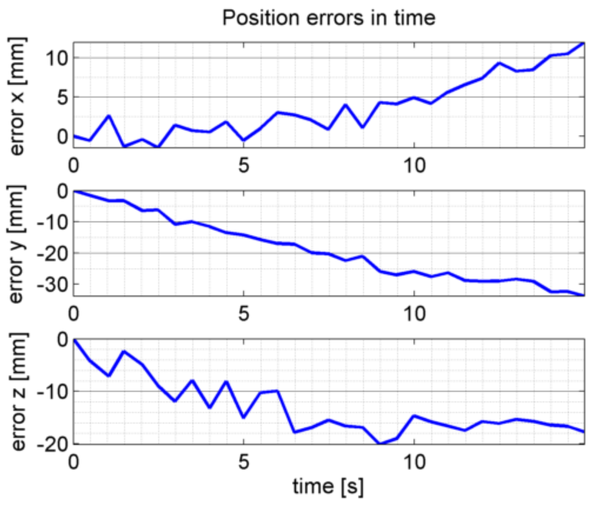
(b) CFO



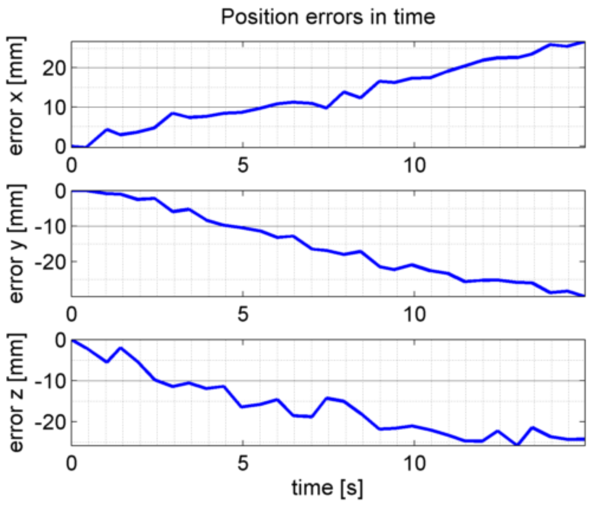
(c) LP

Figure 7.52: Diagonal flight - Position

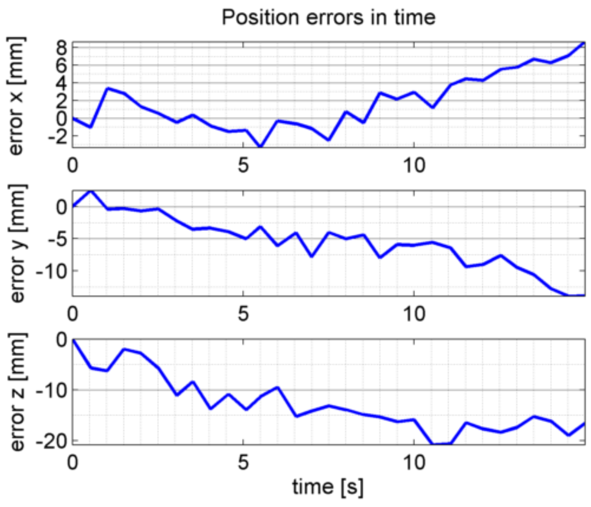
The blue lines represent the PA10 reference while the red circles represent Dokuganryuu reconstruction.



(a) CF



(b) CFO

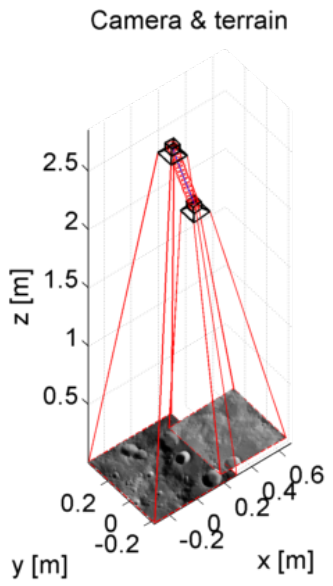


(c) LP

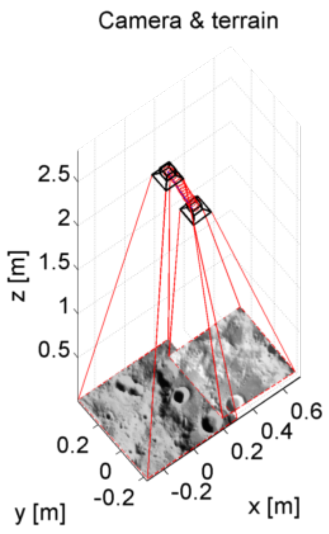
Figure 7.53: Diagonal flight - Position Error

As expected combining two motions result in an higher drift (Figure 7.52 and 7.53). The error in y_{tr} is mostly given by gyro drift while error in the x_{tr} directions can be caused by erroneous distance measurement or by miscalculation in intrinsic camera parameters.

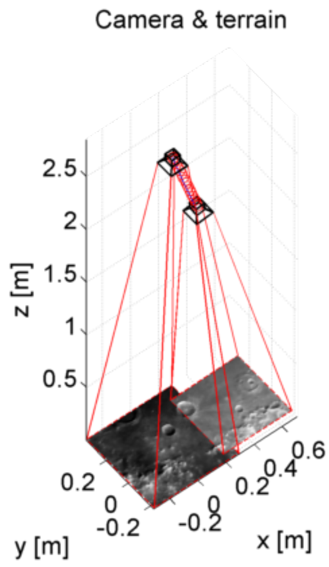
7.3.7 Diagonal flight with pitch



(a) CF



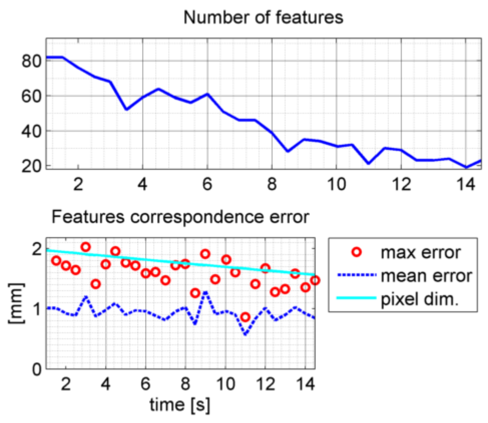
(b) CFO



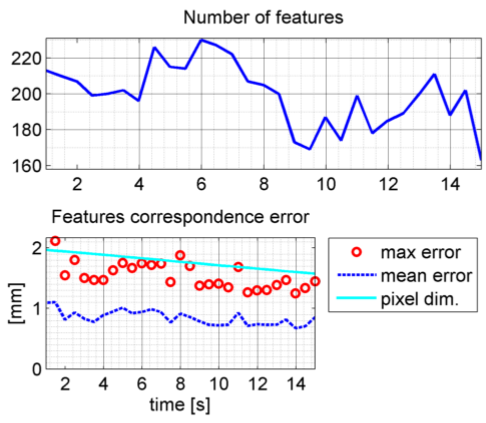
(c) LP

Figure 7.54: Diagonal flight with pitch - Reconstructed trajectory and terrain

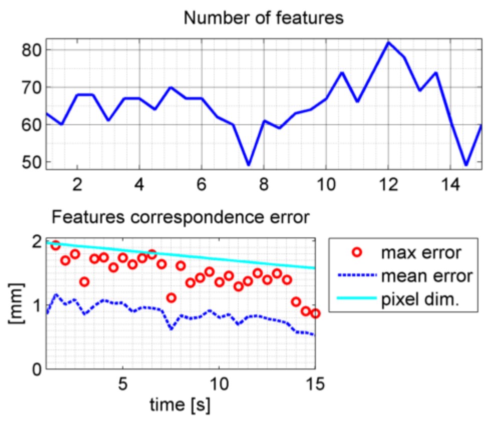
This sequence is similar to the previous but it adds a pitching rotation at the constant rotational speed of 0.5°/s around y_{tr} (Figure 7.54). The combination of two translations, an in plane rotation and the analysis of previous tests suggest an higher drift.



(a) CF



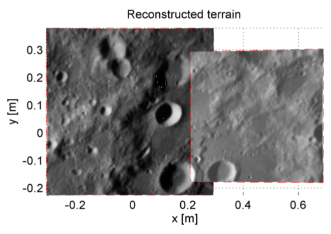
(b) CFO



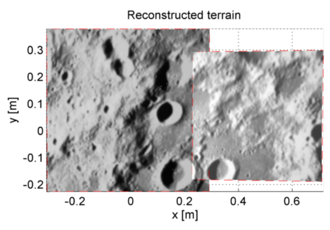
(c) LP

Figure 7.55: Diagonal flight with pitch - Features and error comparison

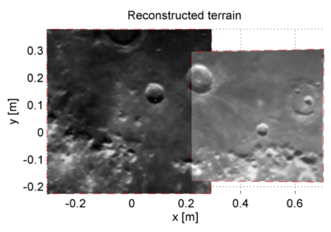
The complex rototranslation motion imposed on the camera involves a relative larger displacement between subsequent images. This involves larger errors in feature matching, as confirmed by Figure 7.55. Despite of this, feature displacement error still remains below the pixel size limit, giving a good landscape reconstruction visible in Figure 7.56.



(a) CF

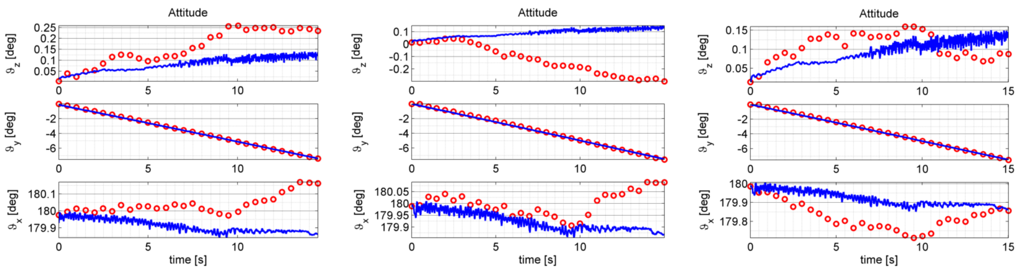


(b) CFO



(c) LP

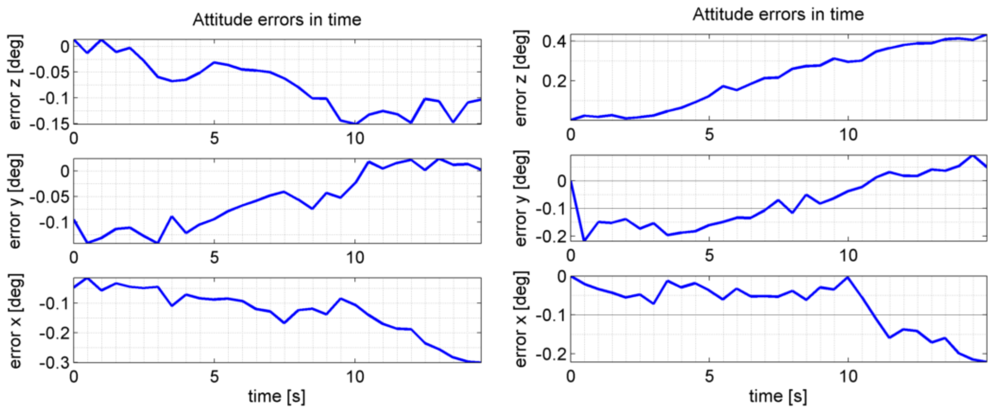
Figure 7.56: Diagonal flight with pitch - Mapped terrain



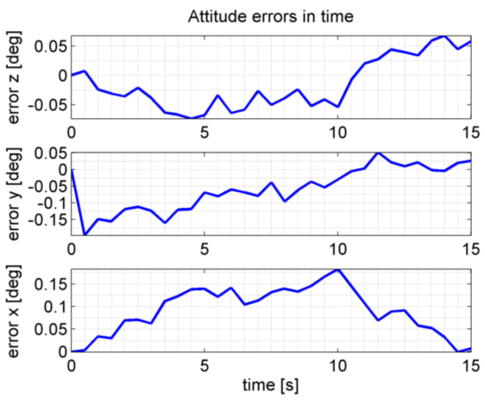
(a) CF (b) CFO (c) LP

Figure 7.57: Diagonal flight with pitch - Attitude (gyroscope)

The blue lines represent the PA10 reference while the red circles represent Dokuganryuu reconstruction.



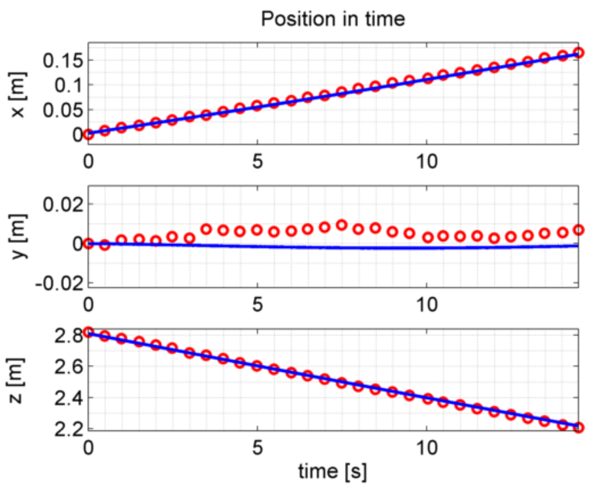
(a) CF (b) CFO



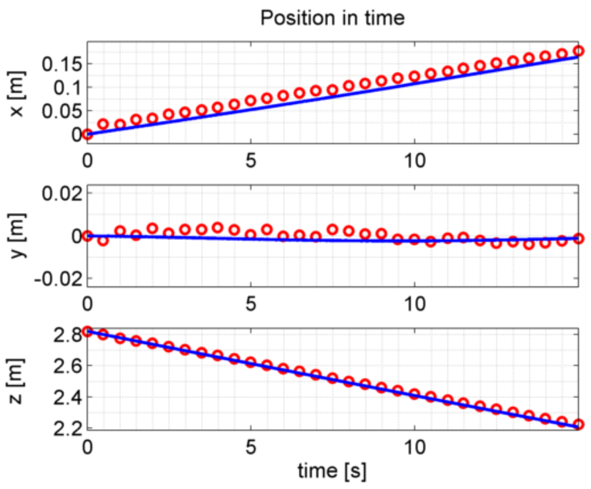
(c) LP

Figure 7.58: Diagonal flight with pitch - Attitude Error (gyroscope)

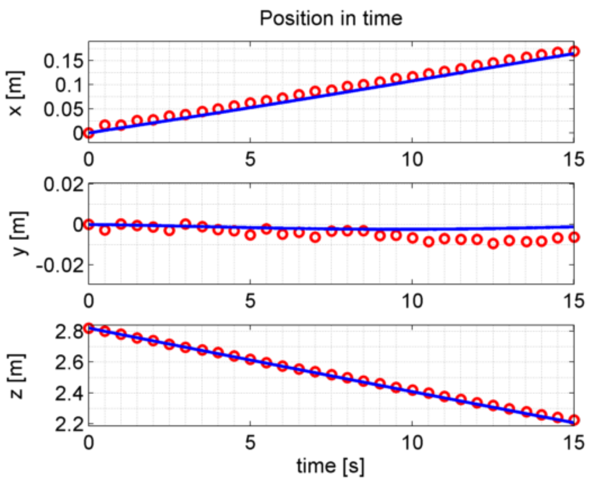
The attitude reconstruction from Figures 7.57 and7.58 is mostly on line with the previous cases, matching the macro rotation and drifting below 0.3 degrees.



(a) CF



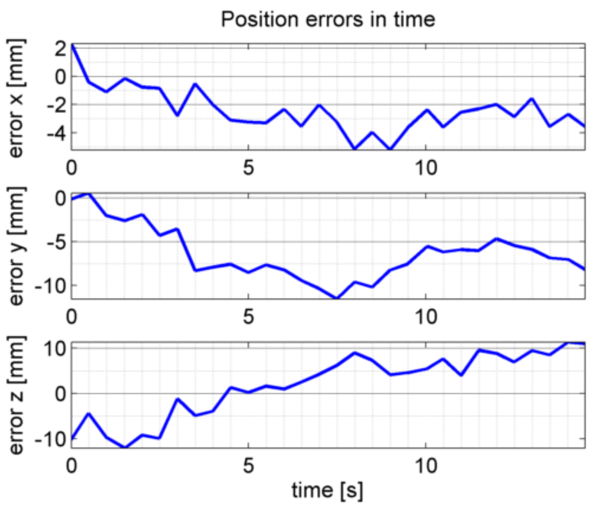
(b) CFO



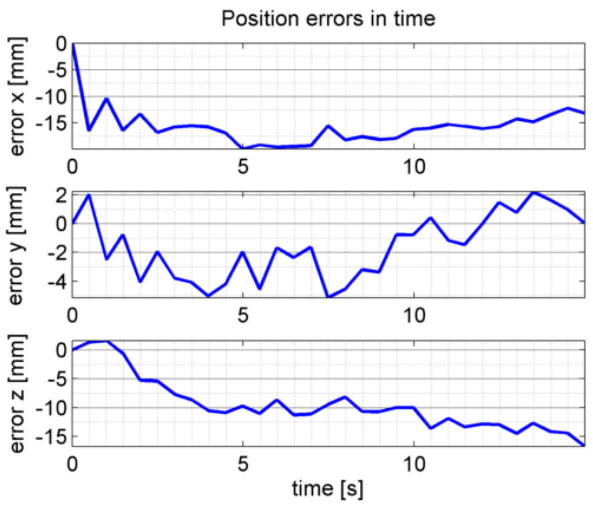
(c) LP

Figure 7.59: Diagonal flight with pitch - Position

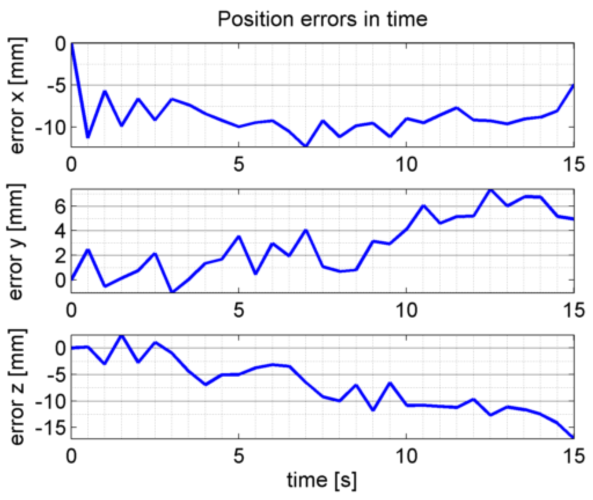
The blue lines represent the PA10 reference while the red circles represent Dokuganryuu reconstruction.



(a) CF



(b) CFO



(c) LP

Figure 7.60: Diagonal flight with pitch - Position Error

It is possible to see in Figure 7.60 that the accuracy in position determination is better than the one obtained without out of plane rotation, despite the more complex motion. In the sub-case of Figure 7.60a an additional position error seems involved. Even so, the drifting behavior of the system is still clearly visible.

7.4 Comparison with previous works

Other works adopting an approach similar to the one presented here are rare in literature; usually rotations are not explicitly addressed, therefore is not easy to confront the problematic here discovered. In [15] a similar testbed is used, but due to weight limitations rotations are not tested, therefore not being able to show this behavior. In many works the vision system is coupled with INS and used to constrain the navigation through epipolar constraint like in [11,16], moreover in [11] rotations are addressed just by gyroscopes and the CV algorithm do not attempt to correct them. Many works in the field use maps to correlate the images or remove the two-view limitations, other fuses with INS with an augmented state vector like SLAM approaches. Here emphasis is put on attaining an algorithm speed capable to work with high constraints.

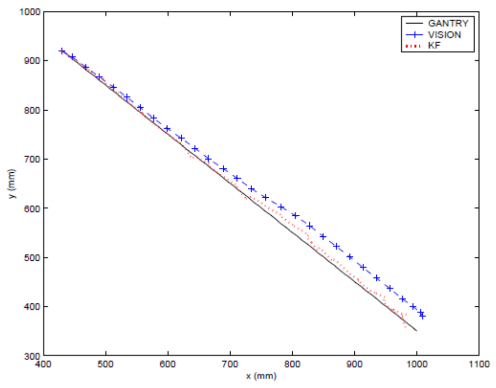


Figure 7.61: Trajectory from Roumeliotis et al.

Some comparison can be made with experimental results given in [15]. In this reference the similar testbed used a 3D mockup of the lunar surface instead of a projector. Plus the IMU involved granted a better accuracy, but at the price of high weight that limited he rotational capabilities of the robotic arm during experiments. Rotations were not involved in that work. Figure 7.61 reports the reconstruction of one trajectory presented in the reference [15]: as in the work presented here, the trajectory given by the robotic arm has been considered as reference and reconstructed by the vision system. Plus, in [15] also a Kalman Filter has been used in order to improve the reconstruction. From the graph, the final error of the vision system seems about 3-5 cm, while the final error in their KF seems about few centimeters.

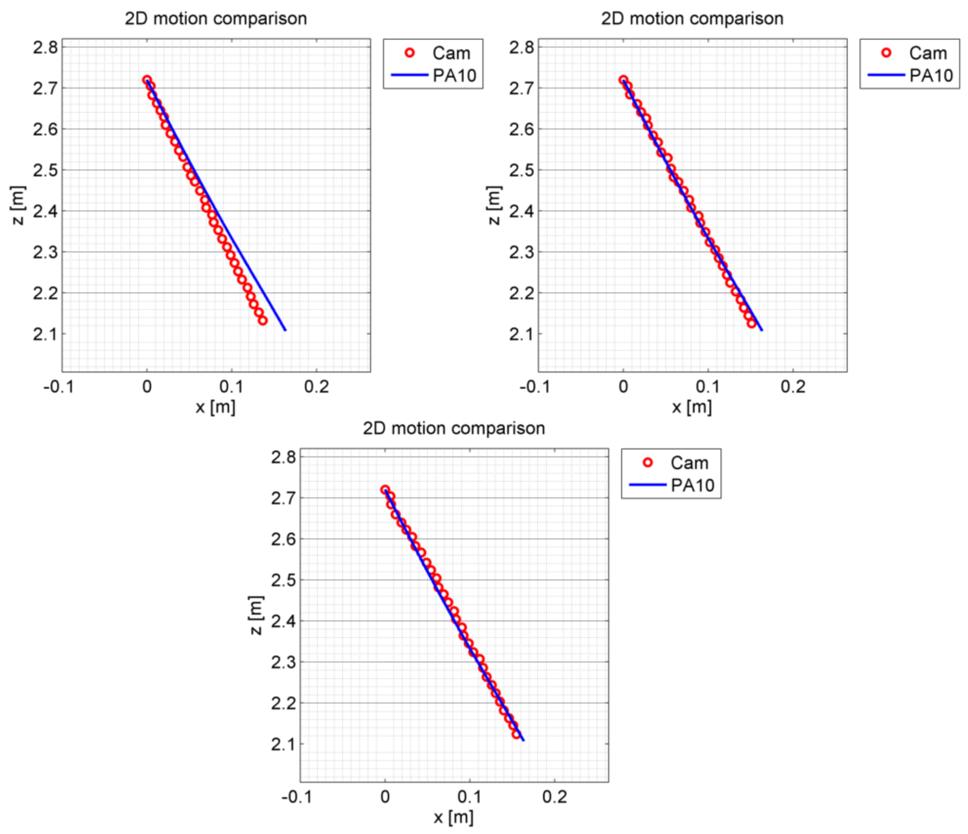


Figure 7.62: Diagonal flight sequences

The blue lines represent the PA10 reference while the red circles represent Dokuganryuu reconstruction.

Figure 7.61 can be compared with the cases with two axis translation seen in 7.3.6 and 7.3.7. Figure 7.62 reports the motion in z_{tr} and x_{tr} for the three sub-cases of Diagonal flight, while Figure 7.63 incorporates also the rotation around y_{tr}. These results have been obtained without using an altimeter but using a good initial positioning to estimate depth and using gyro measures. No filter, beside a low pass filter on IMU measurements, has been used to refine the motion. The results are very similar to the results shown in Figure 7.61 for the pure CV algorithm and this seems a promising beginning to the system improvement: a proper filter can be used to refine the results and fits better the motion.

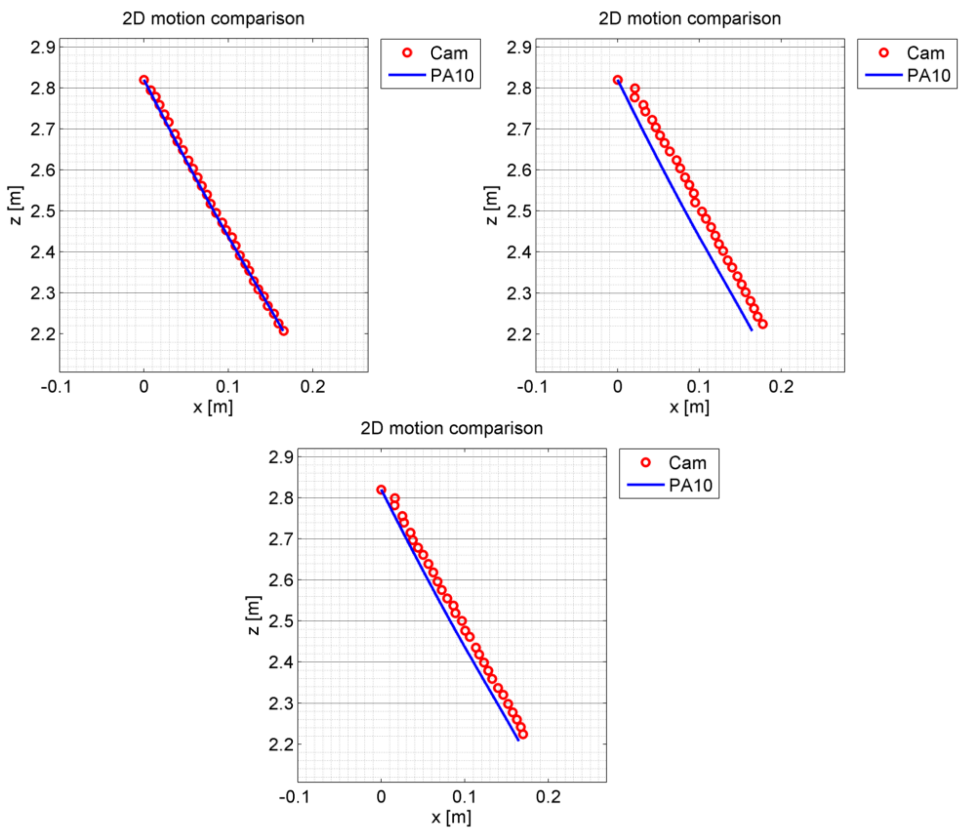


Figure 7.63: Diagonal flight with pitch sequences

The blue lines represent the PA10 reference while the red circles represent Dokuganryuu reconstruction.

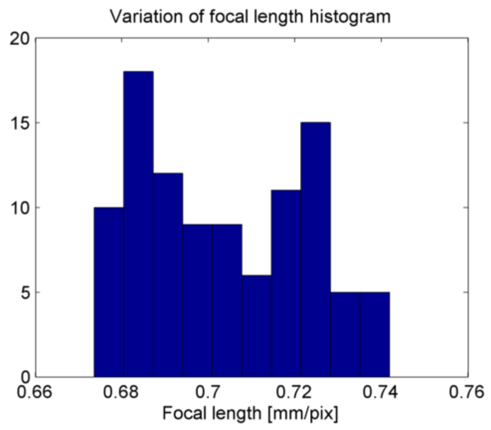
7.5 Algorithm sensitivity

During Dokuganryuu runs some pre-determined values have been considered: in particular camera calibration parameters and initial conditions on position and attitude.

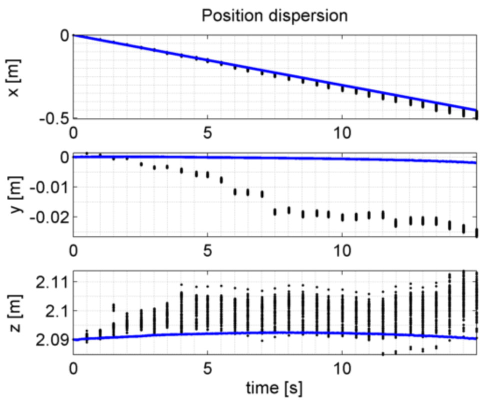
The vision algorithm presented, and validated through tests, uses some pre-determined values to run: in particular the camera calibration and the initial conditions on position and attitude. In this section the sensitivity of such parameters on the final position determination is addressed through a Monte Carlo simulation. The goal of this procedure is not to compute the data dispersion, but to address qualitatively the weight of uncertainties in the final position determination, therefore the number of samples is smaller than usual Monte Carlo application.

7.5.1 Focal length

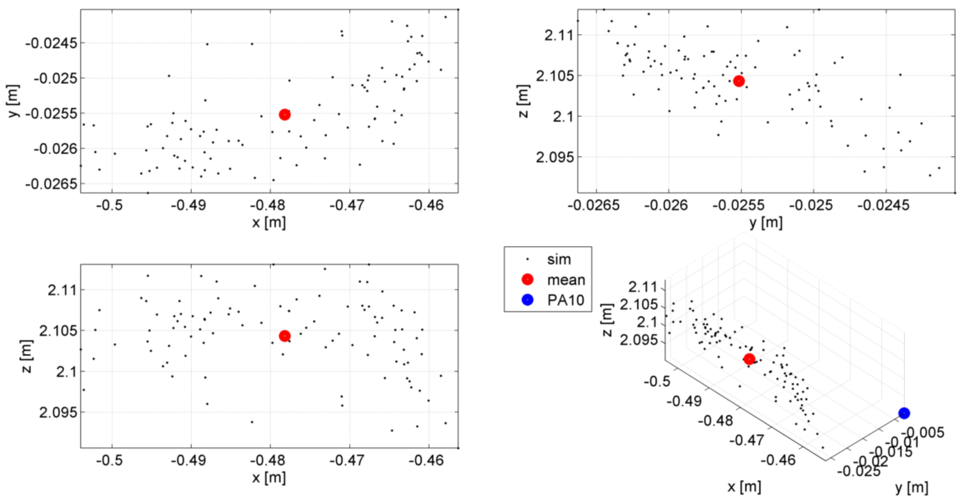
As seen in Section 3.4 the calibration procedure determines the calibration matrix values and in particular the focal distance and the pixel dimensions. In this section it has been considered $\alpha_u = \alpha_v$ (see (3.9) and (3.10)). Since the pixel dimension is fixed a change in these two parameters is a change in focal length. This parameter links the pixel measure to the ratio of in-plane distance from central point and feature depth, therefore it is very important, especially for the in-plane motion. The stochastic simulations were computed with 100 samples varying only the focal length with uniform distribution (5% maximum variation). The simulations uses real gyro output, no correction, of the second pure horizontal motion of Section 7.3.3.



(a) Focal length histogram



(b) Positions

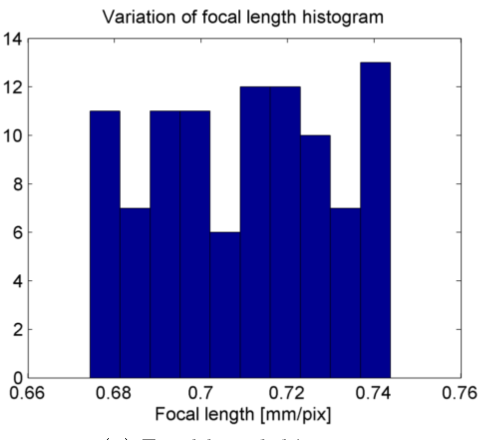


(c) Final point distribution

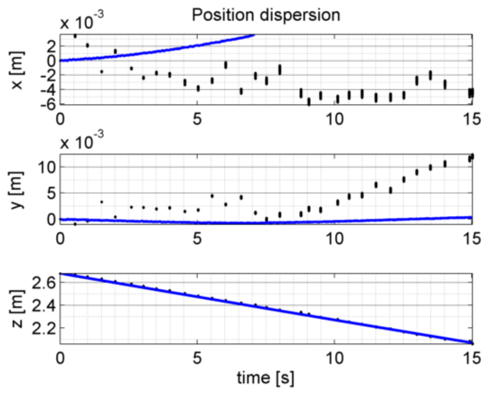
Figure 7.64: Horizontal flight CFO sequence - focal length

Figure 7.64 shows the variation in position estimation due to the uniform distribution of uncertainty in focal length. Such distribution is presented in Figure 7.64a, while Figure 7.64b presents with a blue line the trajectory from the PA10 and with black dots the trajectories determined by Dokuganryuu changing the focal length as mentioned above. Figure 7.64c shows the final landing point distribution in three dimensions.

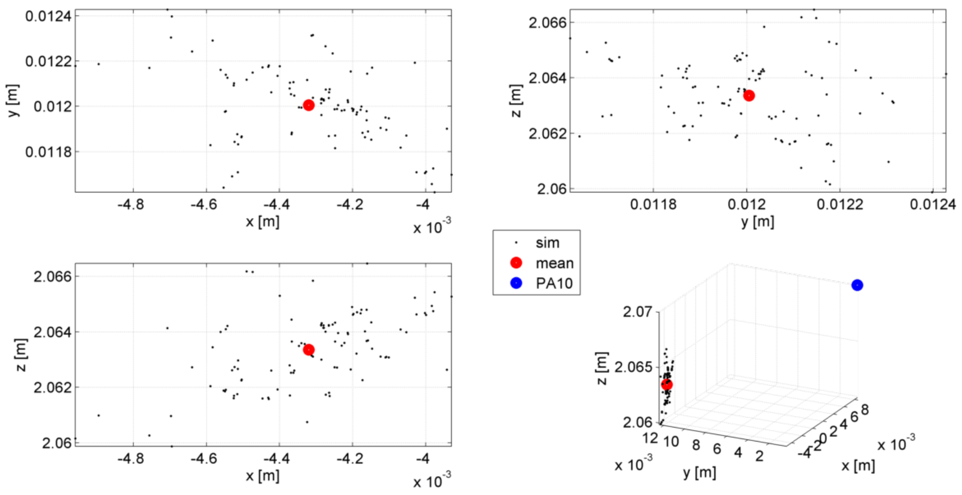
Results clearly indicates that a variation in focal length affect mostly the in-plane motion, therefore affecting the landing point coordinates ($\pm 2\,\text{cm}$ on x_{tr}) more than the distance to the ground ($\pm 0.5\,\text{cm}$ on z_{tr}). This is caused by the sequence higher macro motion on x_{tr}, while on z_{tr} the motion is almost locked. In order to confirm that, a similar simulation has been carried out using the vertical descent sequence where the camera mostly travels on z_{tr}.



(a) Focal length histogram



(b) Positions



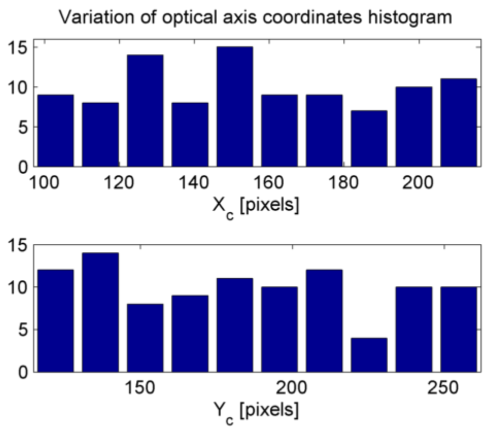
(c) Final point distribution

Figure 7.65: Vertical descent CF sequence - focal length

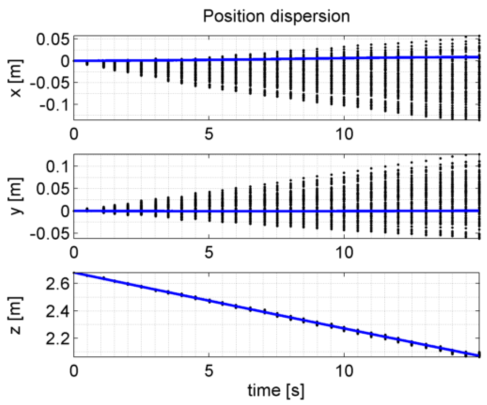
Confronting the distribution of landing points in Figure 7.65 with the ones in Figure 7.64, the variation in all three axes is way lower in this case (± 0.2 cm on z_{tr}, ± 0.02 cm on x_{tr} and y_{tr}). This is a realistic outcome since the depth variation is due to the ratio of pixel coordinates and a small change do not affect much the result. The error is higher in the direction of motion: in the first case on x_{tr} and in the latter case on z_{tr}. The magnitude of the error is higher in motion that deals with plane translation like the one in Figure 7.64.

7.5.2 Optical axis coordinates

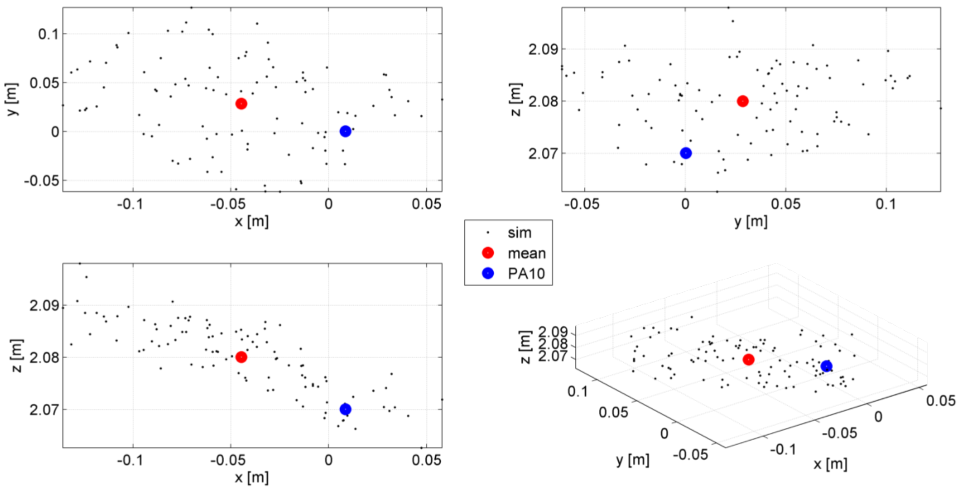
The optical axis coordinates are the other two main parameters given by the calibration. The vertical rotation sequence in 7.3.2 has showed that if the optical axis is not coincident with the rotation axis the apparent error increases. It is checked if a static variation of the reference point, i.e. the optical axis, can reduce the drift seen in these sequences (validating the assumptions) and how heavy that influence is on the determination of the landing site. In the following simulations the optical axis coordinates are changed for each run of Dokuganryuu with a uniform distribution of 100 samples (40% maximum error), shown in Figure 7.66a.



(a) Optical axis histogram



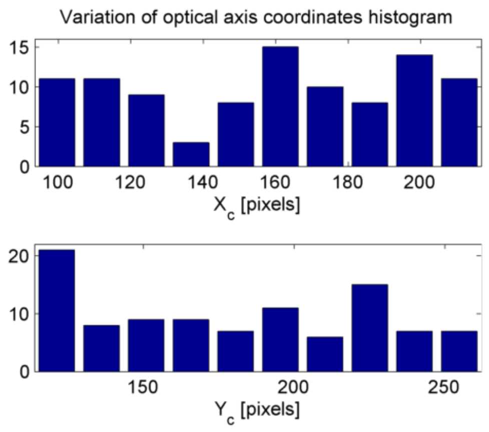
(b) Positions



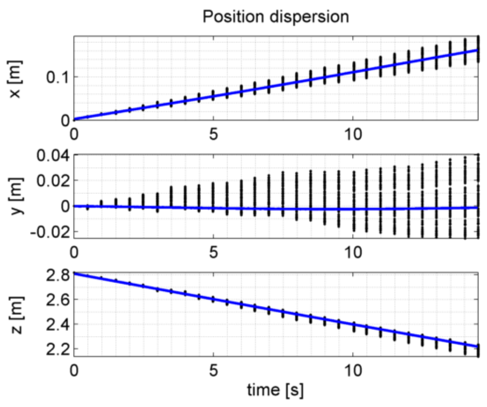
(c) Final point distribution

Figure 7.66: Vertical descent with roll CF sequence

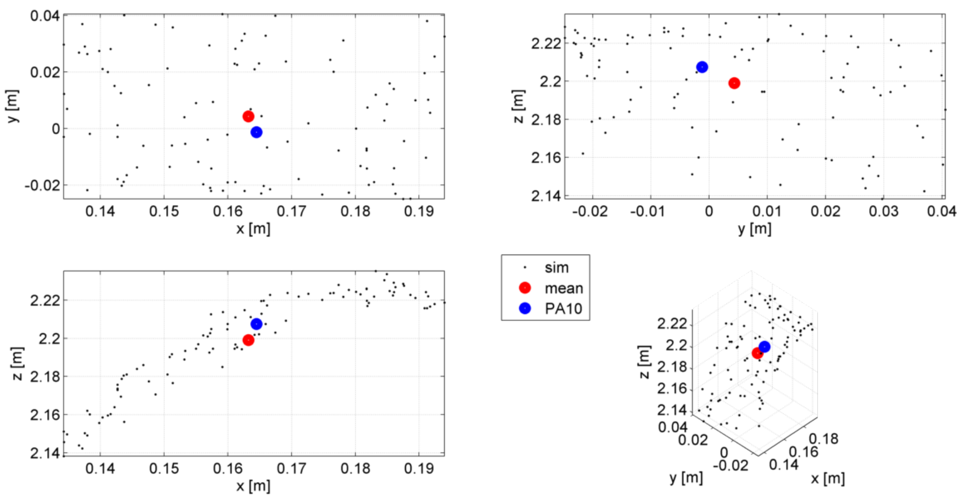
From the simulations in Figure 7.66, the variation in the reference indeed affect greatly the final position, but this is mostly connected to the in-plane rotation. Another interesting fact is that few simulations arrived near the PA10 reference for the final point and as suggested a fixed variation of reference in the image can reduce the apparent drift. Should be noticed that this variation is constant in this case, since the motion and rotation are at constant speed, in a real scenario the rotation axis, i.e. the center of mass, should be addressed in real time since the variation of mass in the lander, the fuel sloshing and other causes may change the center of rotation. The results prove that with little improvement the algorithm might be able to determine eccentricity and thus directly give the vehicle state without further operations.The variation of this reference should also influence other motions that involve rotations. Taking for example the first sequence of Diagonal flight with pitch and applying the same procedure the results are given in Figure 7.67.



(a) Optical axis histogram



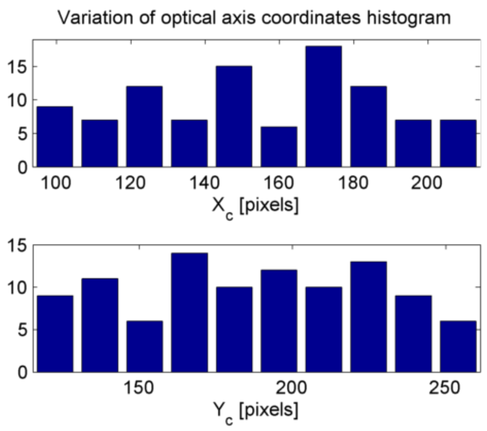
(b) Positions



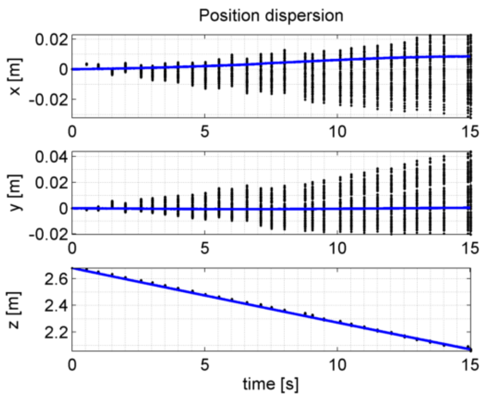
(c) Final point distribution

Figure 7.67: Diagonal flight with pitch CF sequence - optical axis

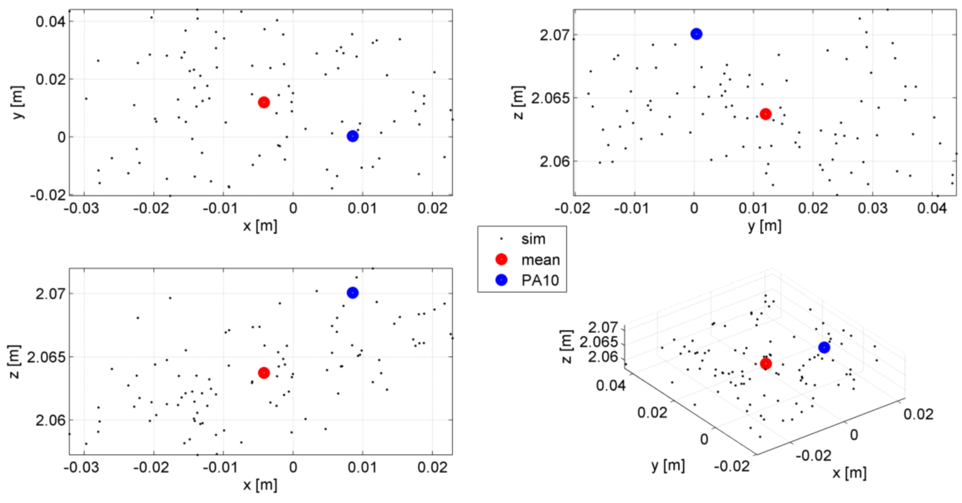
The variation shown in Figure7.67 the dispersion of the final point is not located on the horizontal plane only, but, more generally, it tends to lie in a plane perpendicular to the trajectory. The effect of rotation axis uncertainty may be dramatic.



(a) Optical axis histogram



(b) Positions



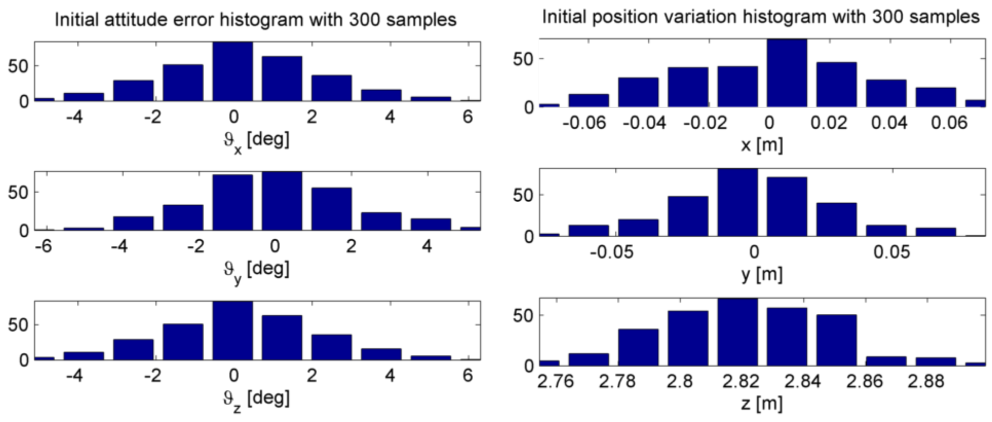
(c) Final point distribution

Figure 7.68: Vertical descent CF sequence - optical axis

In case of pure vertical motion without rotation (Figure 7.68) such uncertainty is less effective. Should be noticed that in these simulation the variation in optical axis position is intentionally larger than the values that could be expected by a wrong calibration, in order to represent uncertainty in the axis of rotation taken as reference.

7.5.3 Initial conditions

Initial conditions are always problematic in INS based navigation, and this of course reflects on the optical navigation algorithm here devised, since it focuses on monocular image sequence and needs an initial estimate in order to reconstruct the full 3-D motion. In order to address better the uncertainty on initial position and attitude, a simulation similar to the previous one is carried out on the Diagonal flight with pitch CF sequence, that excites two translations and one rotation degrees of freedom.



(a) Attitude error histogram (b) Position error histogram

Figure 7.69: Diagonal flight with pitch CF sequence - initial condition error

The initial attitude error and position distributions shown in Figures 7.69a and 7.69b represent the 300 sample used for the simulation. A normal distribution with standard deviation of $\simeq 3\,\text{cm}$ has been used for the position error; a $\simeq 2°$ standard deviation has been used for the normal distribution of the attitude error.

The initial errors in x_{tr} and y_{tr} coordinates are included but the effect on the final result are static since they are not exploited by Dokuganryuu: an initial error in x_{tr} or y_{tr} lead to, at least, the same error in the final point. On the other hand the altitude uncertainty affects the whole reconstruction since it is used as scale factor for the images. The errors in attitude are proven to do the same.

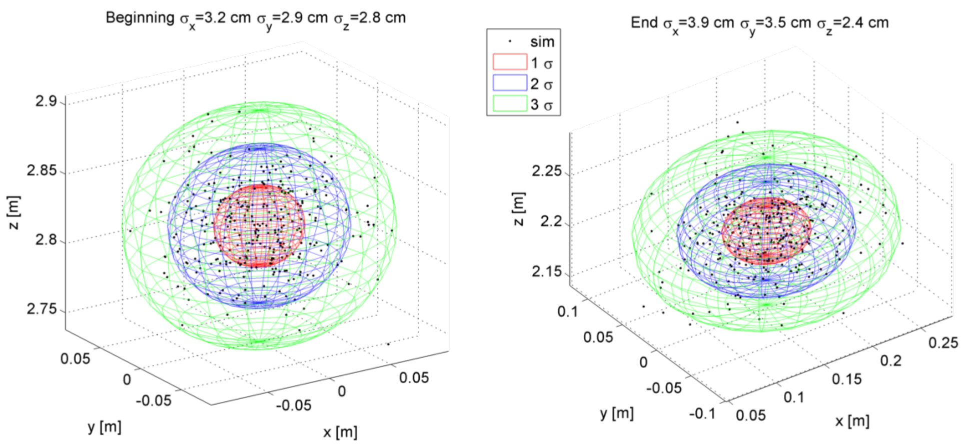
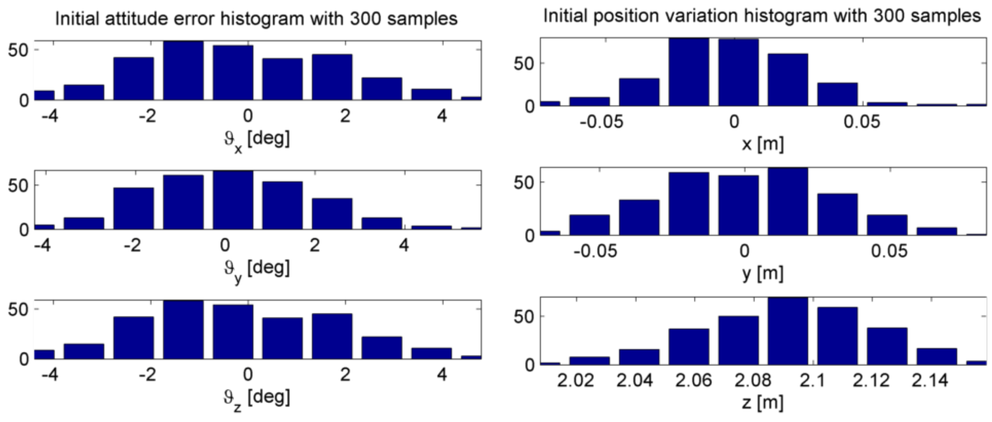


Figure 7.70: Initial and final point distribution of Diagonal flight with pitch - initial condition error

The final point distribution depicted in Figure 7.70 shows that a uniform uncertainty in the three directions leads to a non-uniform distribution at the end of the sequence. In Figure 7.70 it is easy to see that the standard deviation on the z_{tr} axis is reduced while on the other two axes is increased, confirming the divergent nature of the error in x_{tr} and y_{tr} directions. The reduction on z_{tr} is probably due to the intrinsic nature of the algorithm that tends to reduce uncertainty whenever the motion is toward the vertical direction.



(a) Attitude error histogram (b) Position error histogram

Figure 7.71: Horizontal flight CF sequence - initial condition error

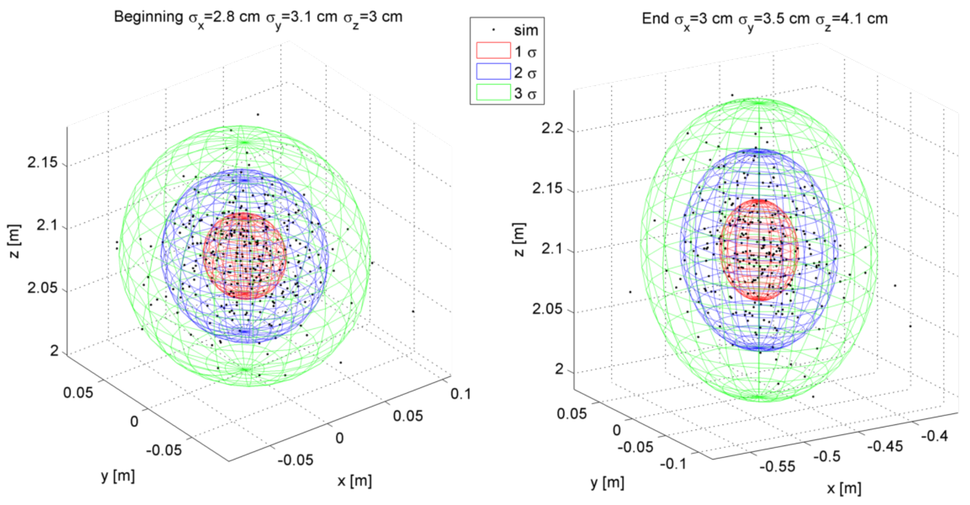
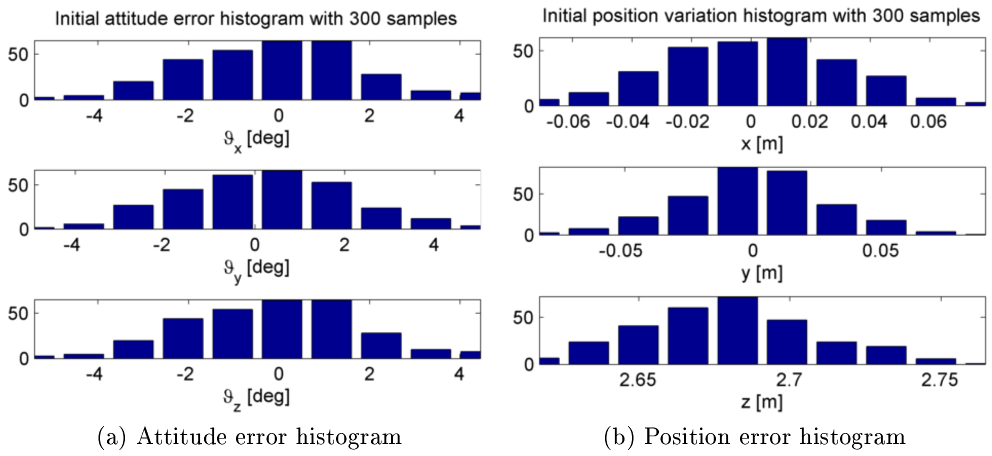


Figure 7.72: Initial and final point distribution of Horizontal flight - initial condition error

With a similar experiment on the Horizontal flight CF sequence, presented in Figures 7.71 and 7.72, it is possible to see that uncertainties in x_{tr} and y_{tr} are almost unchanged, while the uncertainty in z_{tr} increases significantly. This lead to the assumption that z_{tr} uncertainty reduction, seen in Figure 7.70, is deeply connected with the downward motion.



(a) Attitude error histogram (b) Position error histogram

Figure 7.73: Vertical descent CF sequence - initial condition error

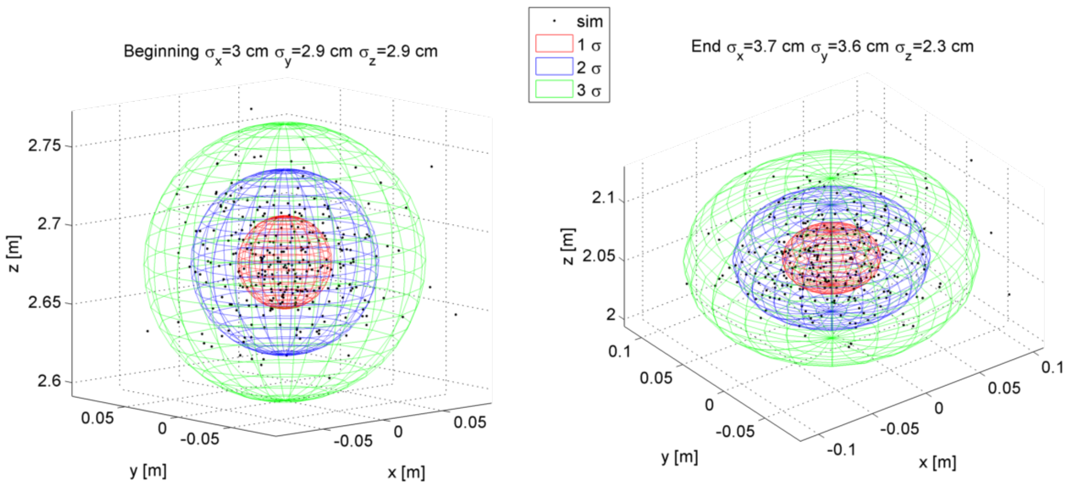


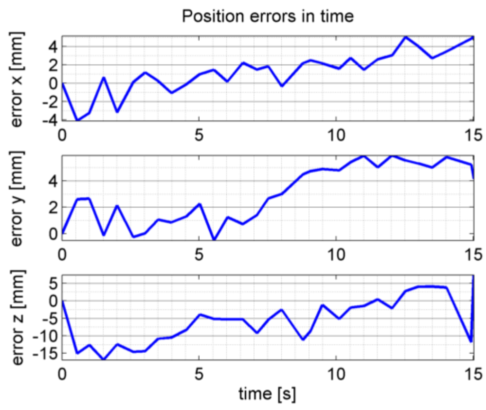
Figure 7.74: Initial and final point distribution of Vertical descent CF - initial condition error

In order to confirm that behavior, another simulation with the Vertical descent sequence has been carried out. Figure 7.74 seems to confirm the speculation of uncertainty reduction on altitude whenever the motion is towards the ground. The uncertainty is however connected with the mean final point instead of the true reference, therefore it is not strictly correct to assume that errors in z_{tr} reduces over time, due to the diverging behavior of the system. What is presented is the behavior of the system that, in case of vertical translation, tends to converge to a point, i.e. a fixed error. Future works will deepen such study.

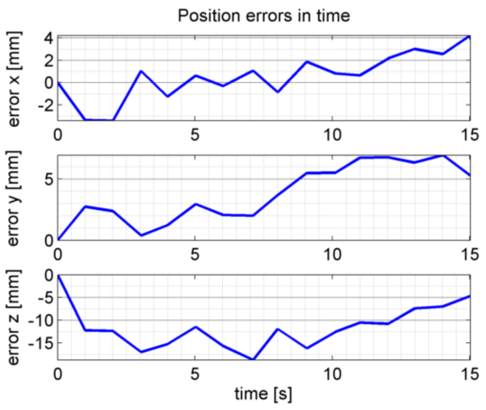
7.5.4 Data rate

Another important setting in the algorithm is the camera data rate. In the experiments above the camera sampled at two Hertz (every 0.5 seconds) and the analysis was carried out with that sequence, however the algorithm may work at different data rates. This analysis allows to to understand better the diverging behavior of the system.

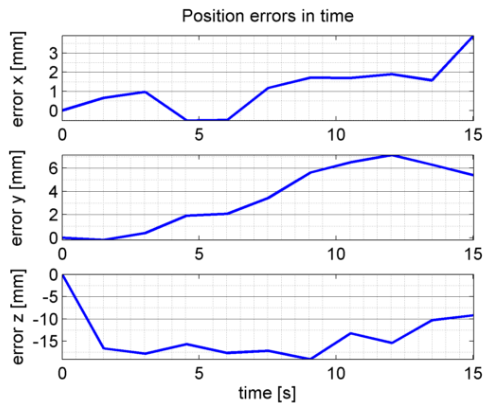
A Vertical descent sequence is analyzed by Dokuganryuu with different frequency updates, varying from 2Hz (0.05s) to 0.33Hz (3s). Figure 7.75 shows the result of these simulations. Apparently a variation in datarate does not affect precision in position reconstruction. Despite of that, an excessive decrease of the computation frequency is not advisable: in fact an increase in the time between consecutive images corresponds to a larger displacement between them. This involves a decrease of the number of matching features available for the computation, that finally could bring a lack of robustness of the navigation algorithm.



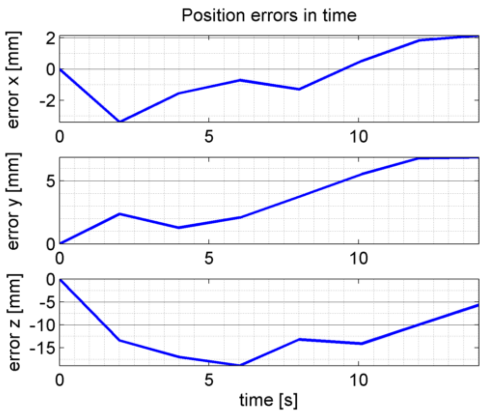
(a) 0.5 s



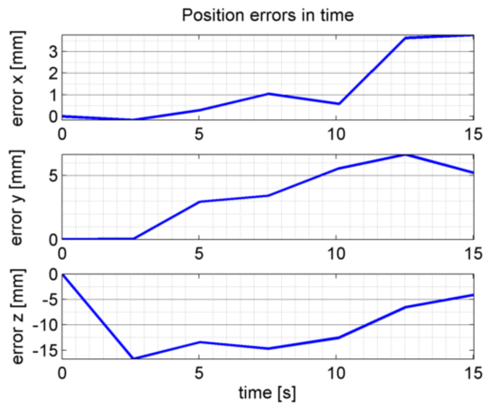
(b) 1 s



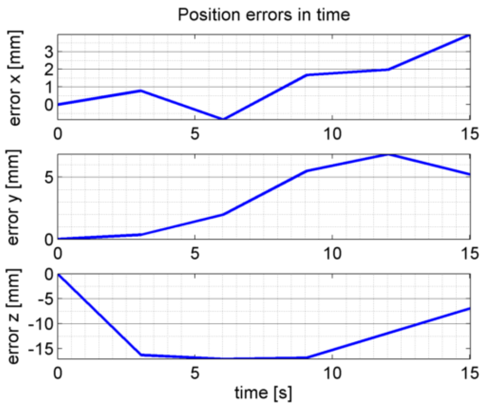
(c) 1.5 s



(d) 2 s



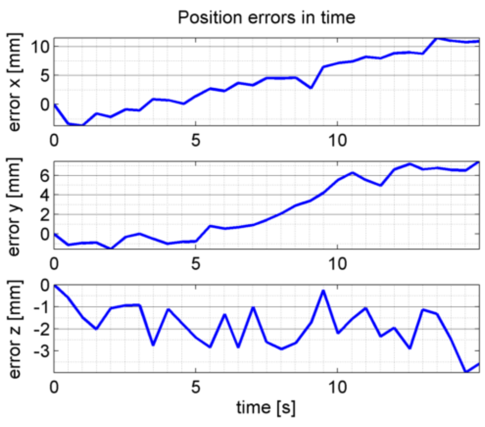
(e) 2.5 s



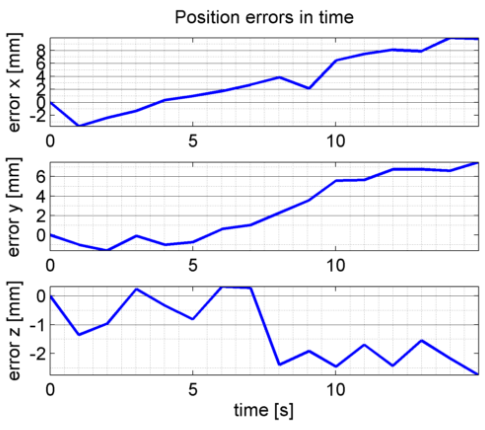
(f) 3 s

Figure 7.75: Vertical descent CF sequence - position error at different data-rates

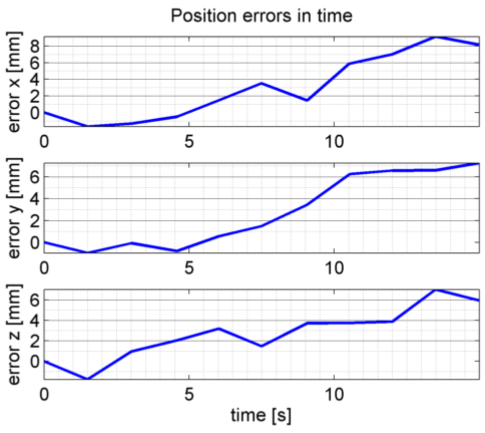
A further verification has been obtained by applying the same analysis to the case of Section 7.3.3 with the CFO terrain. For this trajectory a slower frequency still reduces the amount of matched features but their number remains sufficiently larger. The obtained results, summarized in Figure 7.76, confirm the independence of the accuracy with respect to the update rate. Anyway, more detailed research on the topic should be carried out in future works.



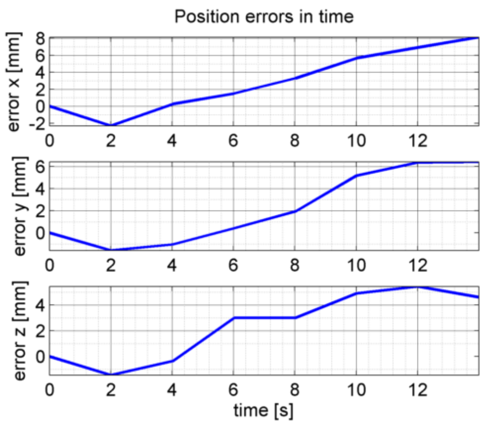
(a) 0.5 s



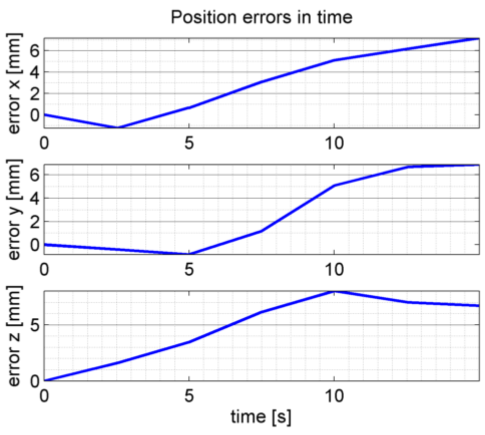
(b) 1 s



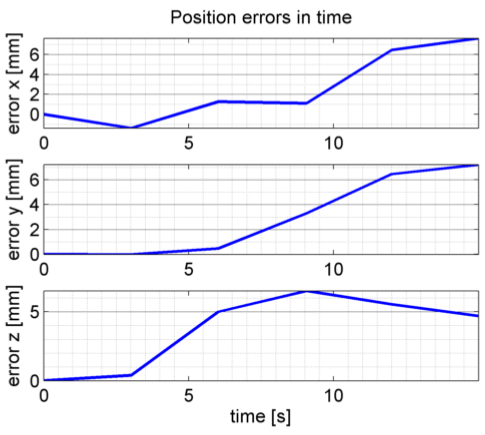
(c) 1.5 s



(d) 2 s



(e) 2.5 s



(f) 3 s

Figure 7.76: Horizontal flight CFO sequence - position error at different data-rates

Chapter 8

Conclusions

In this work a further step in lunar landing navigation has been taken, showing the appeal of a computer vision based state reconstruction. A rather simple navigation algorithm has been devised taking care of the imposed limits:

- the use of low level features ensures robustness and generality,
- low computational complexity is maintained,
- the sensor suite is reduced to a single camera and a 3-axis gyroscope.

Experimental activity proved the inability of a single camera to autonomously reconstruct its position and orientation without external data, moreover in Dokuganryuu the attitude is reconstructed just by gyroscopes, since including the attitude in the LM solver could lead to erroneous position determination. Should be noticed that the algorithm determines relative position and attitude (in the experiments all the equipments were fixed to the ground, therefore corrections due to planet rotation were not considered. In real operational environment the gyroscope signal should be corrected), while it does not explicitly determine velocity and angular velocity, although easily obtainable by differentiation. The algorithm can be easily coupled with a Kalman filter that uses the actual lander motion model to reconstruct better the whole motion.

The level of precision obtained in experimental activities is comparable with previous works, although the computation here is simplified. The precision of the algorithm is surely superior to the INS with less drift alt ought with slower data-rate. The computational speed of the algorithm on real space hardware was not possible, but the algorithm has been designed to be as fast as possible.

Future development

The Dokuganryuu algorithm is just but an initial step in computer vision navigation for lunar landing and can be further optimized and validated. Main critical points that could be improved are:

- LM algorithm improvement, including λ, $v_{\uparrow}$, $v_{\downarrow}$parameter and core modifications to speed up computation
- Model enhancement, including center of mass determination and a more refined motion model
- A more efficient feature matching, by now the task that requires most of the computational effort during each algorithm update.

From the architecture point of view an instrument fusion with altimeter measures seems the most probable combination for this algorithm. Testing INS and Dokuganryuu on a mobile vehicle should answer to the questions left unsolved here. The system could also be improved with a refined model of motion and a Kalman Filter. That approach could be explored in case of a substantial modification in hardware passing from a single camera to a stereo vision: this case seems more appealing and robust, but at the price of an increased computation time.

In case of a stereo-vision application the Dokuganryuu can still be included in the design to improve robustness and computation time. In fact, it has been proven that over short time spans the algorithm does not diverge significantly. Then it can be exploited to estimate the state at high frequency between consecutive update of the slower stereovision, saving computation time. Plus, in case failure of one camera, the navigation could be carried on by Dokuganryuu, still able to work with monocular vision.

Bibliography

[1] NASA, “National space science data center.” Accessed June 2014, from http://nssdc.gsfc.nasa.gov/.

[2] Y. Cheng, A. Johnson, and L. Matthies, “Mer-dimes: a planetary landing application of computer vision,” in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 806–813, IEEE, 2005.

[3] Y. Cheng, J. Goguen, A. Johnson, C. Leger, L. Matthies, M. Martin, and R. Willson, “The mars exploration rovers descent image motion estimation system,” *Intelligent Systems, IEEE*, vol. 19, no. 3, pp. 13–21, 2004.

[4] M. Maimone, A. Johnson, Y. Cheng, R. Willson, and L. Matthies, *Autonomous navigation results from the Mars Exploration Rover (MER) mission*. Springer, 2006.

[5] J. Kawaguschi, A. Fujiwara, and T. Uesugi, “Hayabusa (muses-c) rendezvous and proximity operation,” in *56th International Astronautical Congress*, 2003.

[6] Z. Zexu, W. Weidong, and C. Pingyuan, “An algorithm of asteroid craters detection based on gradient vector flow snake model,” in *Systems and Control in Aeronautics and Astronautics (ISSCAA), 2010 3rd International Symposium on*, pp. 545–549, IEEE, 2010.

[7] Y. Cheng and J. Miller, “Autonomous landmark based spacecraft navigation system,” 2003.

[8] Y. Cheng, A. E. Johnson, L. H. Matthies, and C. F. Olson, “Optical landmark detection for spacecraft navigation,” 2003.

[9] B. Leroy, G. Medioni, E. Johnson, and L. Matthies, “Crater detection for autonomous landing on asteroids,” *Image and Vision Computing*, vol. 19, no. 11, pp. 787–792, 2001.

[10] N. Trawny, A. I. Mourikis, S. I. Roumeliotis, A. E. Johnson, J. Montgomery, A. Ansar, and L. Matthies, “Coupled vision and inertial navigation for pin-point landing,” in *NASA Science and Technology Conference*, 2007.

[11] D. D. Diel, P. DeBitetto, and S. Teller, “Epipolar constraints for vision-aided inertial navigation,” in *Application of Computer Vision, 2005. WACV/MOTIONS’05 Volume 1. Seventh IEEE Workshops on*, vol. 2, pp. 221–228, IEEE, 2005.

[12] M. George and S. Sukkarieh, “Inertial navigation aided by monocular camera observations of unknown features,” in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 3558–3564, IEEE, 2007.

[13] L. Kneip, A. Martinelli, S. Weiss, D. Scaramuzza, and R. Siegwart, “Closed-form solution for absolute scale velocity determination combining inertial measurements and a single feature correspondence,” in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 4546–4553, IEEE, 2011.

[14] A. I. Mourikis and S. I. Roumeliotis, “A multi-state constraint kalman filter for vision-aided inertial navigation,” in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 3565–3572, IEEE, 2007.

[15] S. I. Roumeliotis, A. E. Johnson, and J. F. Montgomery, “Augmenting inertial navigation with image-based motion estimation,” in *Robotics and Automation, 2002. Proceedings. ICRA’02. IEEE International Conference on*, vol. 4, pp. 4326–4333, IEEE, 2002.

[16] D. Zachariah and M. Jansson, “Camera-aided inertial navigation using epipolar points,” in *Position Location and Navigation Symposium (PLANS), 2010 IEEE/ION*, pp. 303–309, IEEE, 2010.

[17] S. I. Roumeliotis, G. S. Sukhatme, and G. A. Bekey, “Circumventing dynamic modeling: Evaluation of the error-state kalman filter applied to mobile robot localization,” in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 2, pp. 1656–1663, IEEE, 1999.

[18] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping: part i,” *Robotics & Automation Magazine, IEEE*, vol. 13, no. 2, pp. 99–110, 2006.

[19] T. Bailey and H. Durrant-Whyte, “Simultaneous localization and mapping (slam): Part ii,” *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[20] L. M. Paz, P. Jensfelt, J. D. Tardos, and J. Neira, “Ekf slam updates in o (n) with divide and conquer slam,” in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 1657–1663, IEEE, 2007.

[21] J. Sola, “Consistency of the monocular ekf-slam algorithm for three different landmark parametrizations,” in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 3513–3518, IEEE, 2010.

[22] T. Suzuki, Y. Amano, and T. Hashizume, “Development of a sift based monocular ekf-slam algorithm for a small unmanned aerial vehicle,” in *SICE Annual Conference (SICE), 2011 Proceedings of*, pp. 1656–1659, IEEE, 2011.

[23] Z. K. Yavuz and S. Yavuz, “Improvement of the measurement update step of ekf-slam,” in *Intelligent Engineering Systems (INES), 2012 IEEE 16th International Conference on*, pp. 61–65, IEEE, 2012.

[24] M. Montemerlo, S. Thrun, and B. Siciliano, *FastSLAM: A scalable method for the simultaneous localization and mapping problem in robotics*, vol. 27. Springer, 2007.

[25] A. Monjazeb, J. Sasiadek, and D. Necsulescu, “Autonomous navigation among large number of nearby landmarks using fastslam and ekf-slam-a comparative study,” in *Methods and Models in Automation and Robotics (MMAR), 2011 16th International Conference on*, pp. 369–374, IEEE, 2011.

[26] E. Wu, L. Zhao, Y. Guo, W. Zhou, and Q. Wang, “Monocular vision slam based on key feature points selection,” in *Information and Automation (ICIA), 2010 IEEE International Conference on*, pp. 1741–1745, IEEE, 2010.

[27] G. Bresson, T. Féraud, R. Aufrere, P. Checchin, and R. Chapuis, “Parsimonious real time monocular slam,” in *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pp. 511–516, IEEE, 2012.

[28] M. Kleinert, C. Ascher, and U. Stilla, “Comparison of inertial mechanization approaches for inertial aided monocular ekf-slam,” in *Information Fusion (FUSION), 2012 15th International Conference on*, pp. 1594–1600, IEEE, 2012.

[29] P. Lunghi, “Robust control for planetary landing maneuvers,” 2013.

[30] J. Delaune, D. De Rosa, and S. Hobbs, “Guidance and control system design for lunar descent and landing,” 2010.

[31] N. Trawny, A. I. Mourikis, S. I. Roumeliotis, A. E. Johnson, and J. F. Montgomery, “Vision-aided inertial navigation for pin-point landing using observations of mapped landmarks,” *Journal of Field Robotics*, vol. 24, no. 5, pp. 357–378, 2007.

[32] F. M. Mirzaei and S. I. Roumeliotis, “A kalman filter-based algorithm for imu-camera calibration: Observability analysis and performance evaluation,” *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 1143–1156, 2008.

[33] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis, "Observability-constrained vision-aided inertial navigation," *University of Minnesota, Dept. of Comp. Sci. & Eng., MARS Lab, Tech. Rep*, vol. 1, 2012.

[34] C. X. Guo and S. I. Roumeliotis, "Imu-rgbd camera 3d pose estimation and extrinsic calibration: Observability analysis and consistency improvement," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 2935–2942, IEEE, 2013.

[35] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis, "Camera-imu-based localization: Observability analysis and consistency improvement," *The International Journal of Robotics Research*, p. 0278364913509675, 2013.

[36] M. S. Nixon and A. S. Aguado, *Feature Extraction & Image Processing for Computer Vision.* Academic Press, 2012.

[37] Z. Zhang, "A flexible new technique for camera calibration," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 11, pp. 1330–1334, 2000.

[38] C. Minwalla, E. Shen, P. Thomas, and R. Hornsey, "Correlation-based measurements of camera magnification and scale factor," *Sensors Journal, IEEE*, vol. 9, no. 6, pp. 699–706, 2009.

[39] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[40] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision–ECCV 2006*, pp. 404–417, Springer, 2006.

[41] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[42] J. Lewis, "Fast template matching," in *Vision Interface*, vol. 95, pp. 15–19, 1995.

[43] J. Lewis, "Fast normalized cross-correlation," in *Vision interface*, vol. 10, pp. 120–123, 1995.

[44] A. E. Johnson, A. Ansar, L. H. Matthies, N. Trawny, A. I. Mourikis, and S. I. Roumeliotis, "A general approach to terrain relative navigation for planetary landing," in *AIAA Aerospace@ Infotech Conf., Rohnert Park, CA*, 2007.

[45] R. I. Hartley, "In defense of the eight-point algorithm," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 6, pp. 580–593, 1997.

[46] H. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms, MA Fischler and O. Firschein, eds*, pp. 61–62, 1987.

[47] R. Y. Tsai and T. S. Huang, "Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 1, pp. 13–27, 1984.

[48] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[49] P. H. Torr and D. W. Murray, "The development and comparison of robust methods for estimating the fundamental matrix," *International journal of computer vision*, vol. 24, no. 3, pp. 271–300, 1997.

[50] T. Botterill, S. Mills, and R. Green, "Refining essential matrix estimates from ransac,"

[51] M. Zuliani, "Ransac for dummies," *With examples using the RANSAC toolbox for Matlab and more*, 2009.

[52] P. H. Torr and A. Zisserman, "Mlesac: A new robust estimator with application to estimating image geometry," *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 138–156, 2000.

[53] R. A. Maronna and V. J. Yohai, "Robust estimation of multivariate location and scatter," *Encyclopedia of Statistical Sciences*, 1998.

[54] H. Gavin, "The levenberg-marquardt method for nonlinear least squares curve-fitting problems," *Department of Civil and Environmental Engineering, Duke University*, 2011.

[55] H. O. Hartley, "The modified gauss-newton method for the fitting of non-linear regression functions by least squares," *Technometrics*, vol. 3, no. 2, pp. 269–280, 1961.

[56] K. Levenberg, "A method for the solution of certain problems in least squares," *Quarterly of applied mathematics*, vol. 2, pp. 164–168, 1944.

[57] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[58] H. B. Nielsen, "Damping parameter in marquardt's method," tech. rep., Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 1999.

[59] B. P. Flannery, W. H. Press, S. A. Teukolsky, and W. Vetterling, “Numerical recipes in c,” *Press Syndicate of the University of Cambridge, New York*, 1992.