



CRANFIELD UNIVERSITY

Mónica Estébanez Camarena

Visual / LiDAR Relative Navigation for Space Applications

MSc by Research

Autonomous Systems Signals and Autonomy Group

Primary supervisor : Dr. Antonio F. Scannapieco

Secondary supervisor: Dr. Lounis Chermak

October 2018

Acknowledgements

I would like to thank my supervisor, Antonio Scannapieco, for his support and determination during the development of this dissertation and during the whole duration of my MSc by Research. Even before being appointed as my supervisor he always worked side by side with me and helped me in everything he could. When he was assigned the role of my primary supervisor he took the place with determination and made this dissertation, together with two conference papers result of this research and to be published in the Proceedings of the 2018 International Astronautical Congress, possible.

I also want to thank my secondary supervisor, Lounis Chermak, for his constant motivation. I have always found support in him and that has been essential for the development of my MS by Research.

Special mention also has to be made to Luke Feetham who has offered his time and tireless dedication to help me during my research and during the production of this dissertation.

Finally, I would like to thank my friends at Cranfield University, who have made my stay at Cranfield much more enjoyable. Special mention to Marco Zaccaria Di Fraia and Akhil Kallepalli, who pushed me and motivated me when I needed it the most.

The completion of this dissertation would not have been possible with the contribution of all of them.

Abstract

Nowadays, robotic systems are shifting towards increased autonomy. This is the case of autonomous navigation, which has been widely studied in literature and extensively implemented for ground applications, for example on cars or motorbike. However, autonomous navigation in space poses a number of additional and different constraints, e.g. reduced number of features, limited power and processing capabilities and life-cycle, among others, that differentiate the problem of that of the ground.

In this framework, the I3DS Integrated 3D Sensors project intends to propose a solution for autonomous operations in space. I3DS is a joint venture between Cranfield University, Thales Alenia Space and other industrial European partners. The ambition of I3DS is to produce a standardised modular Inspector Sensor Suite (INSES) for autonomous orbital and planetary applications for future space missions. The project is co-founded under the Horizon 2020 EU research and development program and is part of the Strategic Research Cluster on Space Robotics Technologies.

The goal for space applications is hence to develop a LiDAR- and visual-based navigation solution able to estimate the relative pose, i.e. position and orientation, of a non-cooperative target in orbit with respect to the chaser satellite. The navigation solution also encompasses a dedicated and application-oriented pre-processing of the raw data. This work aims to respond to this need by assessing the suitability and limitations of the different pre-processing and navigation algorithms for relative navigation on both the on-board computer and FPGA in space, given the particular and specific constraints imposed by the space environment.

The data generated by the sensors require specific pre-processing in order to be converted into an optimum format for the posterior navigation algorithms. In particular, image pre-processing includes spatial and spectral corrections, while LiDAR data pre-processing comprises point cloud downsizing and outlier removal. Therefore, in order to properly simulate the I3DS INSES, FPGA and on-board computer (OBC) have been considered as hardware solutions to run the algorithms. The OBC has been simulated using a standard desktop computer on which LiDAR codes have been tested, whereas the FPGA has been simulated with the Xilinx UltraZed-EG FPGA board for image pre-processing.

Different algorithms have been tested and tuned to achieve the navigation solution. ICP (Iterative Closest Point), GICP (Generalised Iterative Closest Point), TICP (Trimmed Iterative Closest Point) and a Kalman filter-based registration using a Histogram of Distances descriptor have been evaluated for LiDAR navigation. Stereo-based visual odometry and monocular navigation based on fiducial markers on the surface of the target satellite represented the solutions for visual navigation.

Experimental tests on simulated data showed good results in terms of accuracy, with an error on position below 5% for all the sensors. However, the computational load on the FPGA board should be further optimised. Possible avenues, such as parallelisation on one or multiple FPGA boards, further optimisation of the algorithms and decreasing the acquisition frequency as a last resource are finally discussed.

Table of contents

List of acronyms	4
------------------------	---

Chapter 1 – Introduction

1.1. Autonomous relative navigation in space	6
1.2. I3DS Integrated 3D Sensors project	7
1.2.1. Introduction to I3DS.....	7
1.2.2. I3DS Sensor Suite structure.....	8
1.2.3. I3DS Sensors: LiDAR, stereo camera and high-resolution camera.....	9
1.2.4. I3DS sensor suite use-cases	13
1.3. Introduction to this work	14
1.4. Conclusions	15

Chapter 2 – Introduction to sensors and hardware

2.1. LiDAR sensor	16
2.1.1. LiDAR working principles and general characteristics.....	16
2.1.2. LiDAR sensors in space	18
2.2. High-resolution monocular camera	19
2.2.1. Monocular camera working principles and general characteristics	19
2.2.2. Monocular cameras in space	20
2.3. Stereo camera	21
2.3.1. Stereo camera working principles and general characteristics	21
2.3.2. Stereo cameras in space	22
2.4. FPGA technology.....	22
2.4.1. Introduction to FPGA	22
2.4.2. FPGAs in space	22
2.5. Conclusions	23

Chapter 3 – Pre-processing algorithms

3.1. Preamble: Camera fundamentals	24
3.1.1. Image formation	24
3.1.2. Camera parameters	26
3.2. Image processing	29
3.2.1. Sources of distortion in an image	30
3.2.1.1. Seidel aberrations	30
3.2.1.2. Chromatic aberration.....	32

3.2.1.3. Vignetting.....	32
3.2.2. Image distortion correction methods	35
3.2.2.1. Vignetting correction	35
3.2.2.2. Lens distortion correction	36
3.2.2.3. Histogram equalisation	37
3.2.2.4. Bilateral filtering	39
3.2.3. Other image processing algorithms: Stereo images	40
3.2.3.1. Stereo rectification	40
3.2.3.2. Sparse stereo matching	42
3.2.3.3. Sparse stereo point cloud production	44
3.2.3.4. Dense stereo matching / disparity map production	45
3.2.3.5. Dense depth map / point cloud production	46
3.3. Point cloud pre-processing	46
3.3.1. Sources of distortion	46
3.3.2. Outlier removal	47
3.3.3. Point cloud down-sampling: Uniform sampling	48
3.4. Conclusions	48

Chapter 4 – Navigation algorithms

4.1. Stereo navigation	50
4.2. Monocular navigation with fiducial markers	52
4.3. Point cloud navigation	54
4.3.1. Iterative Closest Point	54
4.3.2. Generalized Iterative Closest Point.....	57
4.3.3. Trimmed Iterative Closest Point	59
4.3.4. Histogram of Distances point descriptor with Kalman Filter-based registration	61
4.3.4.1. Histogram of Distances (HoD) descriptor	61
4.3.4.2. Point cloud matching	62
4.3.4.3. General Kalman filter	62
4.3.4.4. Application of the Kalman filter to point cloud registration.....	65
4.4. Conclusions	66

Chapter 5 – Tests and results

5.1. Data.....	67
5.2. Hardware solutions for data processing	69
5.3. Development environment.....	70
5.3.1. OpenCV libraries	71

5.3.2. Point Cloud Libraries (PCL).....	71
5.3.3. PetaLinux.....	72
5.3.4. Xilinx Software Development Kit (XSDK).....	72
5.3.5. Minicom	73
5.4. Test bench setup design	74
5.5. LiDAR processing results	76
5.5.1. Point cloud pre-processing	76
5.5.2. Point cloud-based navigation	78
5.6. Image processing results.....	85
5.6.1. Image pre-processing.....	86
5.6.2. High-resolution-camera-based navigation	88
5.6.3. Stereo-camera-based navigation.....	90
5.7. Overall computing times.....	99
5.8. Conclusions	100
 Chapter 6 – Conclusions and future work.....	 101
 References	 103
ANNEX - Experimental work	

List of acronyms

ADR	Active Debris Removal
ASIC	Application Specific Integrated Circuit
CCD	Charge Coupled Device
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central Processing Unit
COTS	Commercial Off-the-Shelf Component
EM	Electromagnetic (spectrum)
ESA	European Space Agency
EU	European Union
FPA	Focal Array Plane
FPGA	Field-Programmable Gate Array
FEE	Front-End Electronics
FOV	Field of View
GEO	Geostationary Orbit
GICP	Generalized Iterative Closest Point
GPS	Global Positioning System
HoD	Histogram of Distances
I3DS	Integrated 3D Sensors
ICP	Iterative Closest Point
ICU	Interface Control Unit
IDE	Integrated Development Interface
IMU	Inertial Measurement Unit
INSES	Inspector Sensor Suite
IOD	In-Orbit Demonstrator
JTAG	Joint Test Action Group
KLT	Kanade-Lucas-Tomasi

LEO	Low-Earth Orbit
LiDAR	Light Detection and Ranging
MPSoC	Multiprocessor System on Chip
NASA	National Aeronautics and Space Administration
NIR	Near-Infrared Reflectance
OBC	On-board Computer
OOS	On-Orbit Servicing
PC	Point Cloud
PCL	Point Cloud Library
R&D	Rendezvous and Docking
SDK	Software Development Kit
SEU	Single Event Upset
SVD	Singular Value Decomposition
TAS	Thales Alenia Space
TAS - F	Thales Alenia Space France
ToF	Time of Flight
TrICP	Trimmed Iterative Closest Point
XSDK	Xilinx Software Development Kit

Chapter 1 – Introduction

With a current trend to a higher level of autonomy in robotic systems, the pressing threat of space debris for operational and future missions and planetary exploration becoming more feasible than ever before, research efforts towards an automated robust solution for spacecraft relative navigation is necessary.

The goal of this Chapter is to set the context for a multi-sensor, end-to-end solution for relative navigation for space applications, for the specific scenario of a space rendezvous with a non-cooperative target. Firstly, the general problem of autonomous relative navigation in space will be introduced. Secondly, the I3DS (Integrated 3D Sensors project), within which this work has been developed, will be presented. This will include a general statement of the I3DS project, the system design, the sensors used and the targeted use-cases. Finally, the last section will briefly introduce the research aim, methodology and structure of this dissertation.

1.1. Autonomous relative navigation in space

Currently, there is a trend in robotic systems towards increased levels of autonomy. Such is the case of autonomous navigation, which has been widely studied in the literature and extensively implemented for ground applications, for example on cars and motorbikes, as well as for aerial applications such as drones.

Relative navigation is the problem of finding the relative position and orientation of two objects, one of them being the “target” and the other one, the autonomous platform itself. For space applications, relative navigation refers to the specific problems of determining the relative pose of a target satellite or a celestial body with respect to the chaser satellite, on which sensors will be mounted.

The most relevant applications of this technology are rendezvous and docking with another spaceship, on-orbit servicing and repair (OOS) [1] [2] and active debris removal (ADR) [3] [4]. The problem of rendezvous and docking (R&D) dates back to the NASA Gemini program in the 1960’s, that served as a testbed for the later Apollo program. Contrary to the US manual approach to R&D, the Soviet Union achieved the first autonomous R&D in 1967, between two unmanned Cosmos spacecrafts. Within NASA the need for autonomous R&D began to arise in view of autonomous resupply missions to the International Space Station and autonomous manned and unmanned missions to and from Mars [5].

Although significant advancement has been made from those early days, an increased level of autonomy in rendezvous and docking operations is still needed for future space missions. For example, as stated in [6], for a rendezvous happening at significant distances from the Earth, such as in the OSIRIS-REx mission, the time that a signal would take to be transmitted to the spacecraft creates the need for an autonomous operation. Furthermore, autonomous rendezvous could also lower the operational costs of a space mission [6].

At the same time, the ever-growing space debris situation is a pressing threat for all operative satellites currently in orbit as well as the services they deliver on the ground, and for future space missions. Currently, out of the 4700 satellites in space, only 1800 are still operative. Furthermore, it

is estimated that events causing spacecraft fragmentation such as break-ups, explosions and collisions have resulted in about 21000 space debris objects in Earth orbit. Still, that number corresponds to the currently tracked and categorised debris objects, whereas millions of smaller debris objects are estimated to be in orbit [7]. This situation creates the urgent need to develop highly capable autonomous navigation systems able to tackle the problem.

Contrary to relative navigation for terrestrial or aerial applications, relative navigation in space is still in its early stages of development. This is mainly due to the many additional and different constraints that differentiate this problem from that of relative navigation on the ground. Namely, the different environment, the limitation in power and processing capabilities, the lifespan and performance requirements of space and ground or air systems, and the general unavailability of GPS (Global Positioning System) signals above a certain altitude in space.

Firstly, the space and ground physical environment are highly different. A sensor on the ground is surrounded by an immersive environment with multiple features, planes and other physical structures that can serve as references for localisation and mapping. On the contrary, due to the near-emptiness of space a sensor there will be able to locate mostly only the target satellite within a blank, featureless background, which increases the difficulty in successfully performing the relative navigation. Secondly, due to the reduced size and weight of a spacecraft and to its specialised subsystems, that may be very different to those of a robotic system on the ground, their on-board power and processing resources are more limited than on the ground, which restricts the possible data processing and navigation algorithms to use. Thirdly, because normally in-orbit repairs are not feasible, the expected lifespan of a satellite platform and subsystems extends from launch until the mission end-of-life, which generally spans many years. This, together with the harsh space conditions, such as large eclipse/sunlight temperature gradients, vacuum effects and ionizing radiation, requires for space electronics to be more robust and resilient than those on the ground. Finally, the unavailability of GPS signals above a certain altitude precludes its use for relative navigation in space. This unavailability is due to the fact that the GPS system was mainly designed for terrestrial use and thus, its coverage in space cannot always be guaranteed. Although satellites in low Earth orbit (LEO) can make use of this signal to achieve precision navigation, its secure expansion to higher orbits is still under development. An example of this is NASA's GPS receiver Navigator, able to operate in altitudes up to geostationary orbit (GEO) [8]. All these factors create the need for a robust navigation solution, based on different sensor types, able to perform relative navigation without relying on absolute localisation measurements such as GPS.

1.2. I3DS Integrated 3D Sensors project

1.2.1. Introduction to I3DS

I3DS Integrated 3D Sensors is a project with the ambition to produce a standardised modular Inspector Sensor Suite (INSES) for autonomous orbital and planetary applications for future space missions. The project is co-founded under the Horizon 2020 EU research and development program and is part of the Strategic Research Cluster on Space Robotics Technologies as the Operational Grant n°4 among 6 [9].

The project involves ten European partners, including Thales Alenia Space (France, Italy, UK and Spain), SINTEF (Norway), TERMA (Denmark), Cosine (Netherlands), PIAP Space (Poland), HERTZ Systems (Poland), and University of Cranfield (UK) [10].

1.2.2. I3DS Sensor Suite structure

The developed INSES will be formed by different building blocks, each one corresponding to a different sensor, with a common interface. Its main goal is to autonomously recognize its environment and localise itself within it. The sensor suite is designed for two different use-cases, namely, inspection of orbital assets and planetary surface exploration. Figure 1.1 shows the set of sensors comprising the sensor suite. I3DS intends to demonstrate that these two sensor architectures will be sufficient for any future space robotic mission involving exteroceptive and proprioceptive sensors [9].

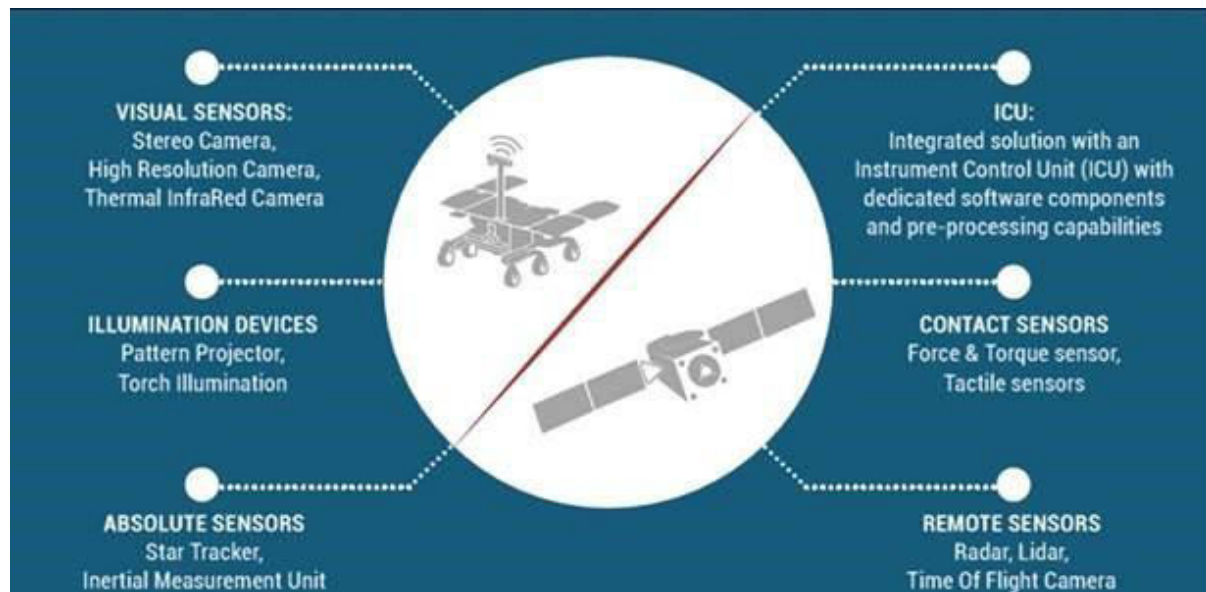


Figure 1.1: Sensors used within the I3DS project for planetary and orbital applications [9].

All sensors are interfaced through SpaceWire networking equipment with an Instrument Control Unit (ICU), in charge of acquiring the data captured by the sensors and pre-processing it. The ICU also has a high-performance multiprocessor system-on-chip (MPSoC) and field-programmable gate array (FPGA) for sensor control, data processing and interfacing with the on-board computer (OBC) over SpaceWire.

The system software components are control and access sensor interfaces, pre-processing of the sensors data and a system interface to communicate with the OBC. All these elements work together to achieve a real-time system.

Finally, in order to be able to achieve a fully operational system in any light condition, means of illumination are added to the sensors suite. This includes a pattern projector and a wide-angle torch. Pattern projection can be used together with the high-resolution camera for navigation purposes, through the creation of 3D point clouds. The wide-angle torch provides general illumination to enable the use of the visual cameras without the satellite being in sunlight. Figure 1.2 shows a block diagram representing the I3DS structure.

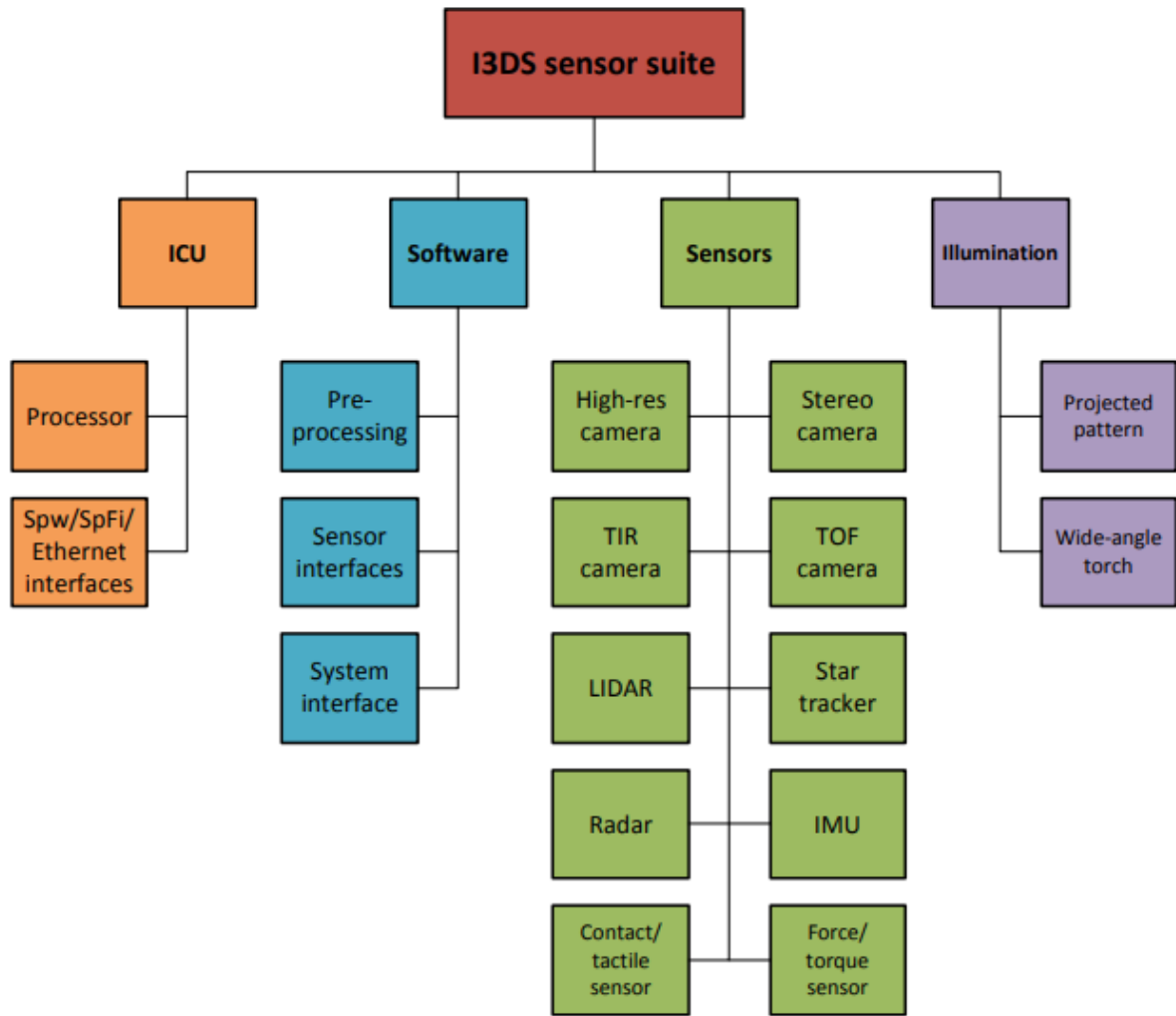


Figure 1.2: I3DS sensor suite block diagram [11].

After the development and testing, the sensors suite will fly as part of the In-Orbit Demonstration (IOD) in the timeframe 2023-2024 [11].

1.2.3. I3DS Sensors: LiDAR, stereo camera and high-resolution camera.

In the framework of the I3DS project, this work consists in the utilisation of a LiDAR, a stereo camera and a high-resolution camera for relative navigation in space. Thus, the focus will be on these three sensors.

The LiDAR sensor used in the I3DS sensor suite is provided by TAS-UK. It is the commercial off-the-shelf (COTS) L3CAM model developed by Beamagine. This choice was based on the fact that the use of a LiDAR was not contemplated in the conception of the I3DS sensor but only incorporated later. The main advantage of this LiDAR is that it does not have any mobile parts, unlike most commercial LiDAR sensors, which makes it more robust than other sensors with rotary parts. This makes the LiDAR more capable of withstanding the high accelerations experienced during satellite launch and of the platform's movements in orbit. Higher robustness also gives it an expected longer lifespan in orbit [11]. The most relevant characteristics of this sensor are presented in Table 1.1 and its functional scheme is depicted in Figure 1.3.

Table 1.1: Technical specifications of the L3CAM LiDAR [12].

	Mid range	Long range
Laser class - Wavelength	Class 1 – NIR	
Range (10% reflectivity)	80 m	300 m
Image resolution	400 x 225 px	250 x 150 px
Field-of-view	80 x 45 °	50 x 30°
Angular resolution	0.2°	
Frame rate	10 frames/s	
Range accuracy (at maximum range)	+/-2 cm	
Size (WxHxD)	150 x 150 x 140 mm	
Weight	1.5 Kg	2.5 Kg
Operating temperature	-40 °C to +65 °C	

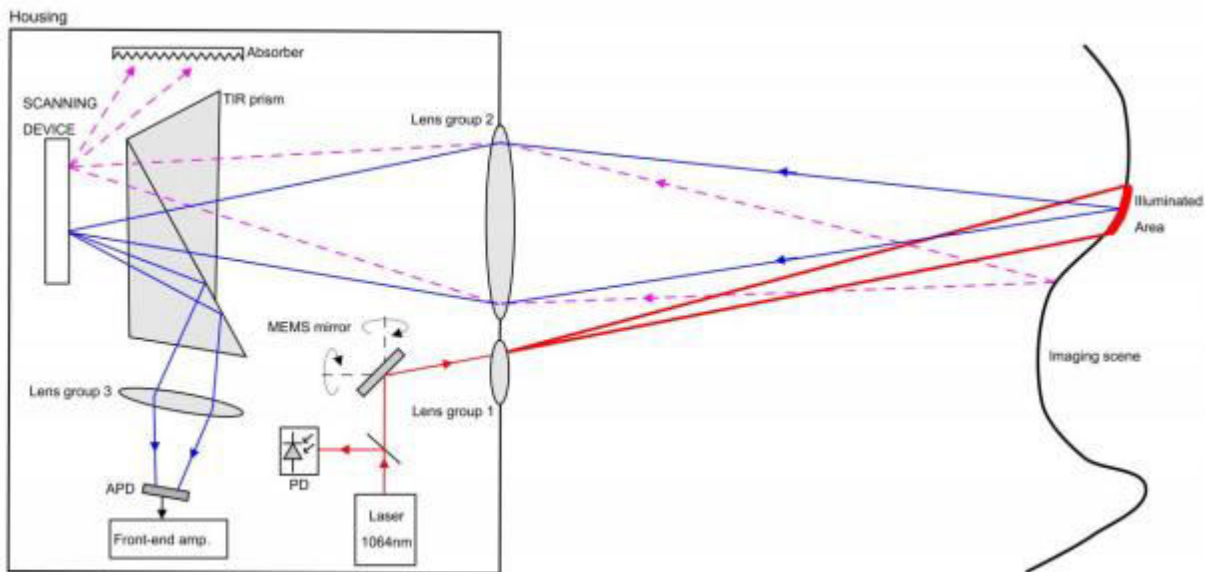


Figure 1.3: Functional scheme of the L3CAM Lidar by Beamagine [11].

The stereo camera in the I3DS sensor suite has been developed by the Dutch company Cosine and consists of an acquisition board and two camera heads. The system has been specifically developed for nanosatellites and comprises a sensor able to sense in the visual and near-infrared regions of the electromagnetic spectrum, read out electronics, data handling and a processing unit [11]. The acquisition board is the interface between the camera heads and the platform and controls the image acquisition. It contains an Altera Cyclone IV FPGA and supports industry-standard Qseven computer-on-module (COM) boards for data processing.

Each camera's Focal Plane Array (FPA) is interfaced to a Front-End Electronics (FEE) that perform the image readout and control of the sensor. After being stored in the FEE, the images are sent to the acquisition board [11]. The camera heads are the Cosine MO6X, whose characteristics are summarised in Table 1.2. Both high-resolution and stereo cameras operate in the visual range of the electromagnetic spectrum and are optimised for operations in the near-infrared spectrum, up to 850 nm. Figure 1.4 shows the block diagram of the stereo camera system used within the I3DS project.

Table 1.2: Technical specifications of the Cosine stereo camera [11] [13].

Intraocular distance	150 mm
Sensing range	VIS / NIR (monochrome)
Focal length	12 mm
Sensor type	CMOS
Sensor model	CMOSIS CMV400
Shutter type	Global
Resolution	4MP - 2048(H) x 2048 (V)
Pixel size	5.5 μm x 5.5 μm
Frame rate	37 fps (12 bit)

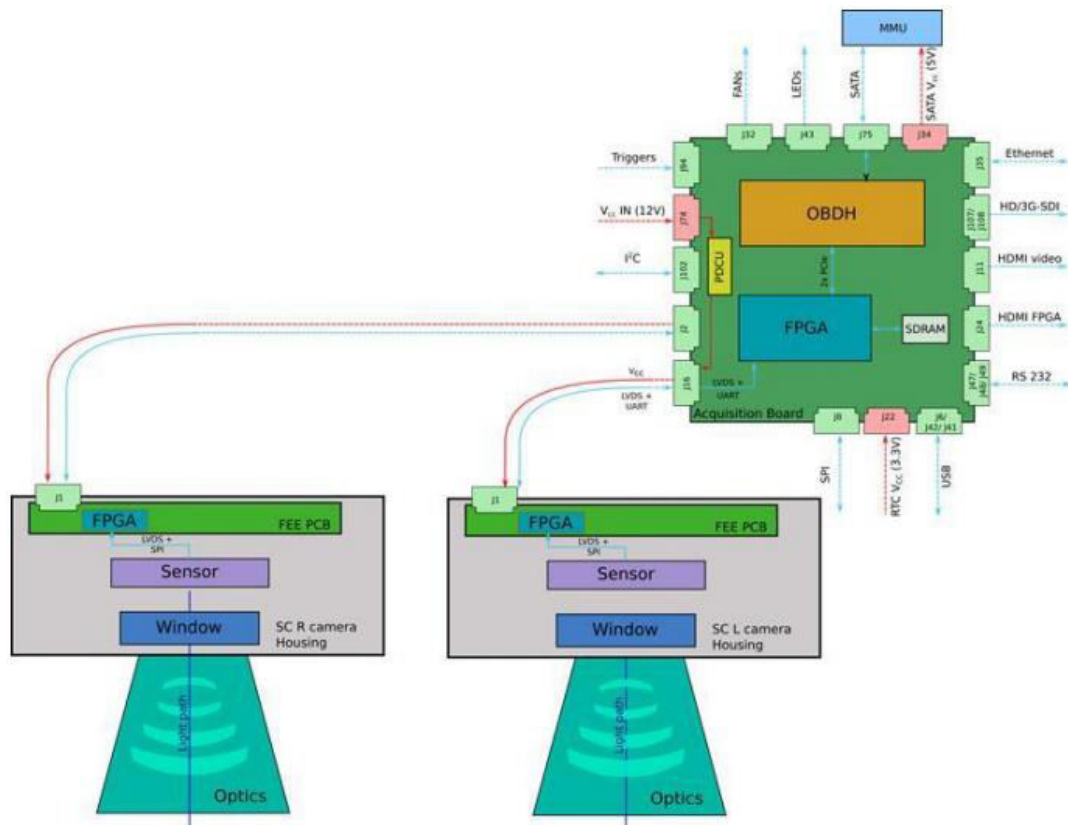


Figure 1.4: Block diagram of the I3DS stereo camera system [11].

Finally, the high-resolution camera has been developed by the Dutch company Cosine and comprises an acquisition board, that is the same as for the stereo camera, and a VIS/NIR camera head. The head, enclosed in its own housing, is composed of the optics, sensor and front-end electronics. As for the stereo heads, the high-resolution camera's FPA is interfaced with the FEE, that carries out operations such as control and readout of the sensor. After storing the image in on-board DDR3 memory, the FEE sends it to the acquisition board. The camera head features a COTS Zeiss Interlock Compact 2.8/21 lens with a 21 mm focal length. As for the stereo camera, the sensor in the camera head is the global shutter CMOSIS CMV400 [11]. The characteristics of the camera are described in Table 1.3. Figure 1.5 represents the I3DS high-resolution camera system, comprising the camera head and the acquisition board.

Table 1.3: Technical specifications of the Cosine high-resolution camera [11] [13].

Sensing range	VIS / NIR (monochrome)
Focal length	21 mm
Sensor type	CMOS
Sensor model	CMOSIS CMV400
Shutter type	Global
Resolution	4MP - 2048(H) x 2048 (V)
Pixel size	5.5 μm x 5.5 μm
Frame rate	37 fps (12 bit)

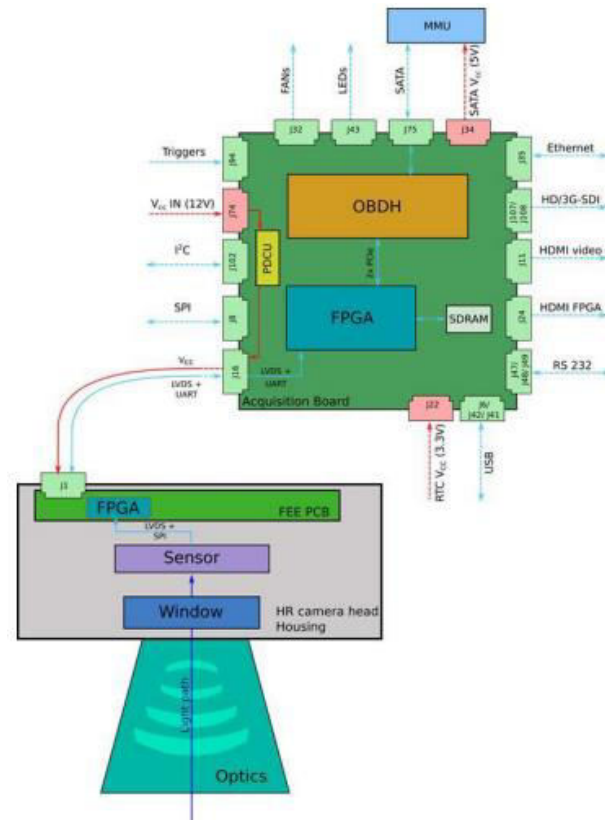


Figure 1.5: Block diagram of the I3DS high-resolution camera system [11].

1.2.4. I3DS sensor suite use-cases

The I3DS sensor suite has been designed to meet the needs of future robotic space missions in two scenarios: orbital and planetary. Orbital applications encompass autonomous rendezvous and capture of a spacecraft, either for on-orbit servicing, structure assembly and cooperative capture (cooperative targets) or for ADR and celestial bodies rendezvous (non-cooperative targets). Planetary applications refer to planetary surface exploration. In both cases, the I3DS suite must be able to autonomously perform relative navigation (i.e. position and orientation determination) with respect to a target body [14].

The I3DS suite for the orbital scenario comprises the following sensors: a star tracker, a stereo camera, a high-resolution camera, a LiDAR and a high frequency radar. For close-proximity operations, the orbital suite also counts with tactile sensors and a force/torque sensor. As for the planetary scenario, the I3DS sensor suite is formed of an inertial measurement unit (IMU), a thermal IR camera, a high-resolution camera and a time-of-flight camera. The illumination in both scenarios is given by the wide-angle torch and the pattern projection. In both cases, the sensor data are processed on the ICU [14].

This work is focused on the orbital use-case. Specifically, the case of ADR of a non-cooperative satellite has been studied. This task can be divided in the phases depicted in Figure 1.6.

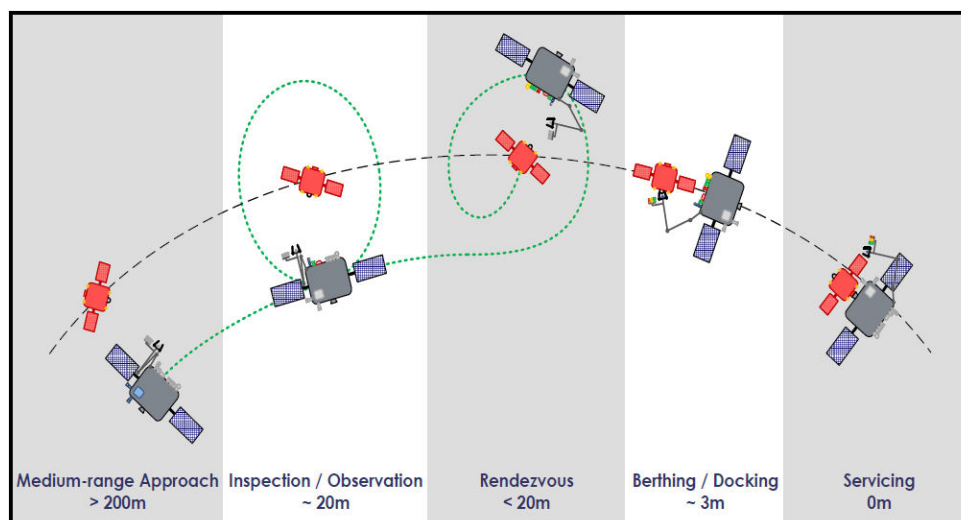


Figure 1.6: Main steps of the orbital use-case on a capture scenario [14].

The particular phase that has been studied in this work is the close-range rendezvous (i.e. relative navigation at a close distance from the target, from a distance of 20 m to 3 m from the target satellite). The precision requirements defined within the I3DS project for this phase of the mission are 0.03 m and 2.5° [14]. Figure 1.7 shows the sensor configuration on-board the servicer spacecraft for the orbital use-case.

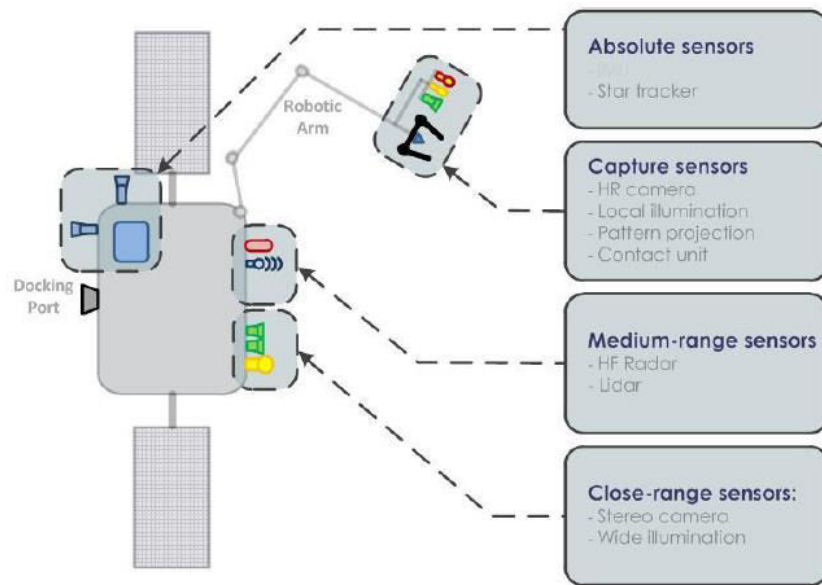


Figure 1.7: Sensor configuration of the servicer for the cooperative capture [14].

1.3. Introduction to this work

Framed within the I3DS project, the aim of this particular work is to develop a relative navigation solution for orbital applications, specifically for the case of a rendezvous with a non-cooperative target, using a LiDAR sensor, a stereo camera and a high-resolution camera. In order to tackle the particular constraints posed by the space environment, the suitability of using an FPGA for part or all the data processing and navigation task is assessed.

After acquisition, the data must first be pre-processed to ensure the provision of robust and optimised inputs to 3-D navigation algorithms. This includes noise reduction and conversion into a format more easily handled by the posterior navigation algorithms. Image pre-processing involves geometric and radiometric corrections on images acquired by each monocular camera, with the stereo camera understood here as a set of two monocular cameras. Additionally, further pre-processing operations are performed on each stereo image pair in order to retrieve depth information and to create point clouds that later can potentially be integrated with data acquired by other sensors such as LiDAR or pattern projection for a fusion-based navigation. Point cloud pre-processing includes data down-sampling with a uniform sampling filter and statistical outlier removal.

Developed navigation algorithms include visual odometry supported by fiducial markers for the high-resolution camera, stereo pairing and Optical Flow for the stereo camera and several different methods for the LiDAR navigation. For LiDAR, the user can choose between the classical Iterative Closest Point (ICP), Generalised ICP, Trimmed ICP and a Kalman filter-based registration method that uses a Histogram of Distances (HoD) point descriptor.

The testing data were simulated by Thales Alenia Space – France and represented a straight-line approach trajectory with the chaser spacecraft moving towards the target spacecraft at a speed of 5 cm/s and from a 20 m distance, without any rotational motion. To be able to perform a comparison between both platforms, both the on-board computer (OBC) and the FPGA board were

simulated in the test-bench. The OBC was represented by a standard desktop computer and the FPGA, by the non-space graded UltraZed-EG FPGA board manufactured by Xilinx. This choice was based on the similarity of this board to other space-graded ones also provided by Xilinx.

This work is structured as follows: First, the main working principles of the considered sensors are introduced in Chapter 2. Chapter 3 and 4 describes the pre-processing and navigation methods mentioned above, respectively. The implementation of the designed software architecture on both the standard desktop computer, simulating the on-board computer, as well as on the FPGA board are described and the obtained results are presented in Chapters 5 and 6. Finally, Chapter VII comprises the conclusions reached by the author as well as a discussion of possible ways to improve the achieved performance together with which methods are promising and which ones should be avoided for relative navigation in space.

1.4. Conclusions

This Chapter has presented the I3DS project within which this work has been developed. I3DS is a project co-funded under Horizon 2020 EU research and development program and part of the Strategic Research Cluster on Space Robotics Technologies. Its main ambition is to produce a standardised modular INSES for autonomous orbital and planetary applications for future space missions.

In this work, a focus has been made on orbital applications. More concretely on the problem of relative navigation for space applications using visual cameras, namely, a high-resolution camera and a stereo camera, and a LiDAR sensor. In this context, relative space navigation refers to the problem of estimating the relative pose (i.e. position and orientation) of a target spacecraft with respect to a chaser spacecraft on which the sensors will be mounted.

The different elements integrating the I3DS sensor suite have been described, with an emphasis on the visual cameras and the LiDAR sensor.

Finally, the main research goals of this MSc by Research as well as the structure of this dissertation have been defined.

Chapter 2 – Introduction to sensors and hardware

Hardware used on-board satellites has to satisfy more constraints than hardware for terrestrial, aerial and maritime application, owing to constraints related to the space environment. Space-qualified hardware needs to be able to withstand intensive mechanical loads during launch and separation, ionizing space radiation, large eclipse/sunlight temperature differences and the extreme-vacuum of outer space. Furthermore, because post-launch, in-orbit repairs are generally not feasible, space hardware also needs to have a long lifespan that can support the mission during its whole operational life.

The aim of this Chapter is to introduce the hardware, including sensors and processing platforms, used in this work. The working principles, general sensor characteristics and a review of their use in space will be presented. In this way, the Chapter will contain the following sections: In Section 2.1., LiDAR sensors will be described. The second and third sections present the visual cameras characteristics, that is, the high-resolution monocular cameras and stereo cameras. Finally, FPGA boards and their use in space are introduced in Section 2.4.

2.1. LiDAR sensor

2.1.1. LiDAR working principles and general characteristics

Light Detection and Ranging (LiDAR) sensors are active remote sensing devices that use light to obtain the range to one or more points on a target object. In space applications, the target objects are commonly the surface of the Earth for remote sensing, and the surface of a planetary body or a target spacecraft for landing and relative navigation applications, respectively. The sensor emits a light beam, generally a laser beam, to measure the distance to the point where it is reflected. Because the speed of light is known, this distance or range can be easily obtained, at least ideally, using the Equation (2.1).

$$r = \frac{ct}{2} \quad (2.1)$$

where r is the range from the sensor to the sensed point on the target object, c is the speed of light, t is the time of flight (ToF) or the time that the laser beam takes to return to the sensor since it was emitted, after being reflected by the target object. Finally, the distance resulting from multiplying these two magnitudes must be divided by 2 to take into account the round-trip of the laser beam (i.e. sent from the sensor to the sensed object and reflected back from the object to the sensor). Different sensors use different ways to measure the laser ToF, including Pulse or Flash sensors, Continuous Wave sensors, Pseudo-Random Number sensors and Compressed Sensing sensors. The precise definition of these sensors is out of the scope of this work but can be found in [15].

Depending on how the laser beam is emitted, LiDAR sensors can be categorised into the following three main types [15]:

- Scanning sensors: These sensors illuminate the scene using only one narrow laser beam and one only detector. The laser beam is swept over the whole Field of View (FOV) through mirrors, lenses and other moving devices. The advantage of these sensors is that because they have only one detector they are relatively easy to calibrate. Furthermore, the laser beam can be pointed very precisely to create very high-resolution point clouds. However, moving devices are the most common cause of hardware failure. Also, the time necessary to create a point cloud of a whole scene is longer than if the scene was illuminated all at once.
- Detector Array sensors: These systems illuminate the whole scene at once using one broad laser beam and an array of detectors to acquire the return signal. On the contrary to scanning sensors, they do not make use of any rotary parts and thus are mechanically more robust. Additionally, the required time to capture a certain scene is shorter. However, the point cloud resolution is lower than for scanning LiDARs.
- Spatial Light Modulator sensors: Still under development, these sensors illuminate parts of the scene in a certain pattern using multiple laser beams but only one detector to measure the return signal.

Figure 2.1 shows the working principle of scanning and detector array LiDAR sensors.

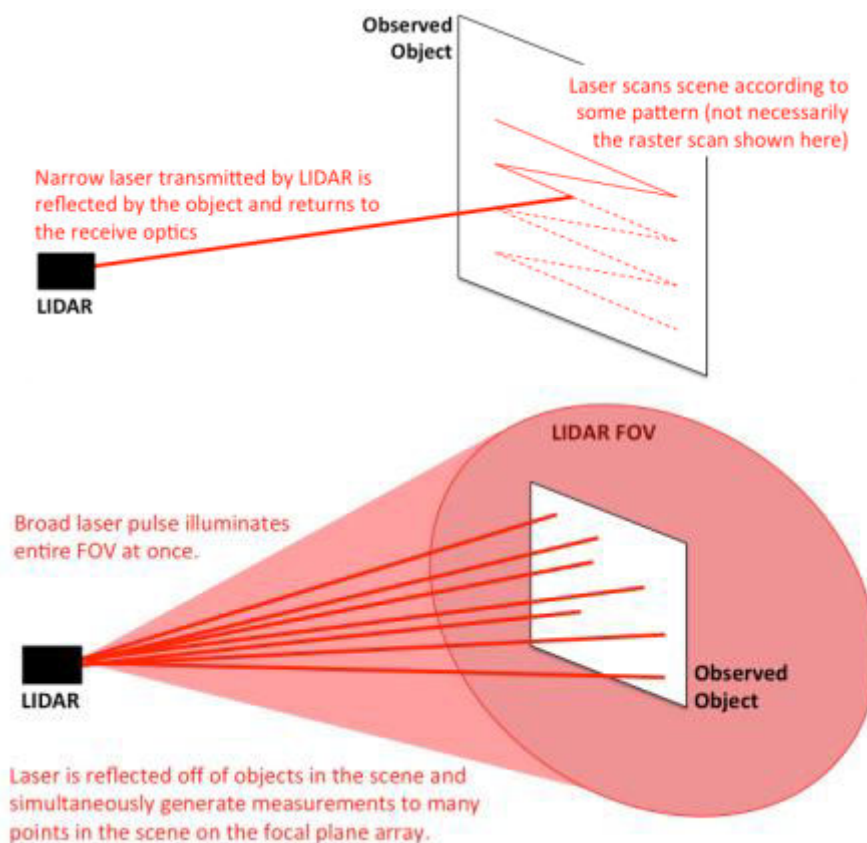


Figure 2.1: Comparison of scanning (top) and detector arrays (bottom) LiDARs [15].

All the signal returns form a point cloud that describes the surface shape of the sensed object. Each point is generally characterized by three coordinates, typically $[x, y, z]$ in a Cartesian three-dimensional coordinate system, as well as other auxiliary attributes such as intensity, return number, number of returns, colour intensity information (RGB), scan angle, scan direction, point source ID or GPS time [16].

There are a wide variety of sensors used for different applications mounted on various platforms, from terrestrial to aerial and maritime and thus their characteristics in terms of weight and power can be very diverse. As an example, the Velodyne's Puck (VLP-16) LiDAR sensor used for the experiments carried out in our laboratory at Cranfield University (see Figure 2.2) weighs 830 grams, features 16 channels, can acquire up to 600 000 points per second, has a 360° Horizontal FOV and a $\pm 15^\circ$ Vertical FOV. Its typical consumption is 8 W and has a 5 Hz – 20 Hz rotation rate. It operates on a wavelength of 903 nm [17]. Figure 2.3 represents an example of a point cloud acquired by this sensor.



Figure 2.2: Velodyne's Puck (VLP-16) LiDAR sensor used for testing at Cranfield University [17].

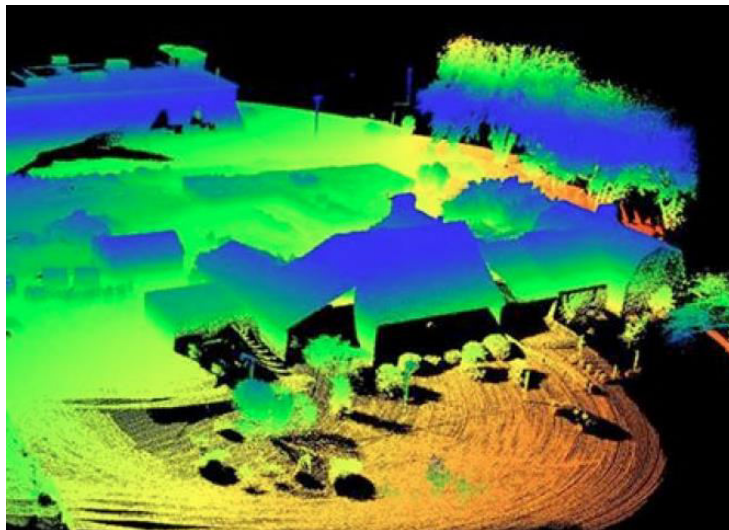


Figure 2.3: Example of a point cloud provided by XactSense [18]. The different colours represent the intensity of the received signal reflected by each point.

2.1.2. LiDAR sensors in space

There are multiple examples of existing and proposed LiDAR space missions, including Earth observation (atmospheric [19], topographic [20] and oceanographic [21] applications), planetary landing and relative navigation in space.

The first LiDAR used in relative navigation was the Trajectory Control Sensor (TCS) used on-board the Space Shuttle for rendezvous with the Mir Space Station, the Hubble Space Telescope (HST), and the

International Space Station (ISS). Its first flight was the STS-71 to Mir in 1995 and its last one was the STS-135 to the ISS in 2011. Other LiDAR sensors used in space have been the Videometer (VDM) on the ESA Automated Transfer Vehicle (ATV) and the sensor flown on JAXA's H-II Transfer Vehicle (HTV) under the name of RendezVous Sensor (RVS) and on ESA's ATV under the name Telegoniometer (TGM) [15].

However, there are some factors that generally limit the use of LiDAR sensors for space applications, namely, their high mass and energy demand compared to other sensors such as cameras. Therefore, their use in space is still in its infancy.

2.2. High-resolution monocular camera

2.2.1. Monocular camera working principles and general characteristics

Whether monocular or stereo, cameras are passive remote sensing devices that capture the light reflected by the sensed objects. Like in the human eye, where an image is formed when the light being reflected by an object strikes the photoreceptors in the retina, an image in a camera is formed when this same light reaches the camera image plane after passing through the camera aperture and the lenses and mirrors that comprise the imaging system (see Figure 2.4).

There are two kinds of sensors: Charge Coupled Devices (CCD) and scientific Complementary Metal Oxide Semiconductors (CMOS or sCMOS). The main difference is how they perform the readout of the data acquired by the detector. While CCDs have only one single Analog-to-Digital Converter (A/D) to which each pixel signal is serially transferred to after image acquisition, CMOS sensors have one A/D for every column of pixels in the image, as shown in Figure 4. This difference results in the CCD frame rate being limited by the rate at which each pixel can be transmitted to the A/D and digitized, thus giving an overall lower frame rate than CMOS, since all pixels must be digitized by the same A/D. On the other hand, CMOS sensors can achieve a higher frame rate due to a smaller number of pixels being transferred to each A/D. However, each image row must be digitized at the same time which produces small time delays between rows [22]. The schematics of both kinds of sensors are represented in Figure 2.5.

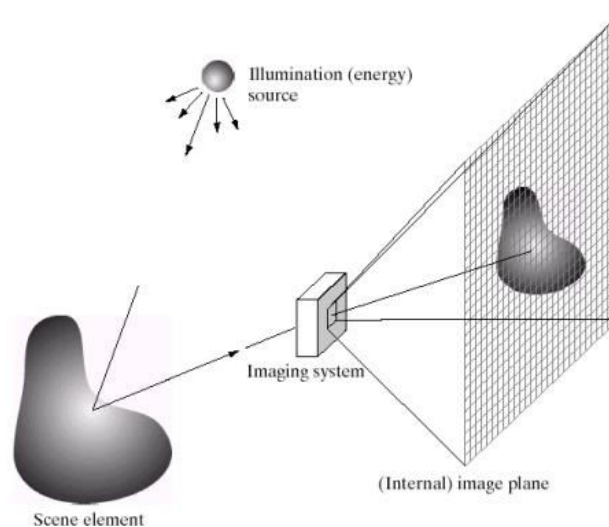


Figure 2.4: Image formation process in a camera [23].

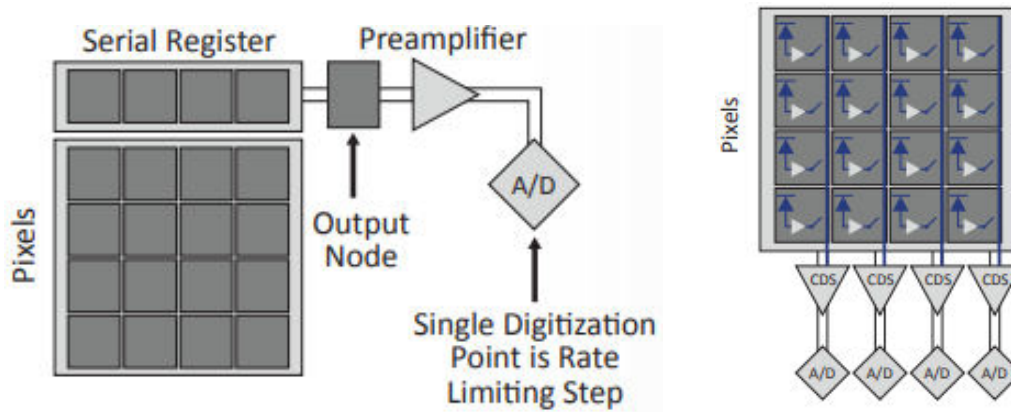


Figure 2.5: Basic schematics of CCD (left) and CMOS (right) sensor architectures [22].

Furthermore, cameras are also characterized by their shutter mode, which can be global shutter or rolling shutter. In the first case, the whole detector is exposed to the incoming light at the same moment in time, which is particularly beneficial when the sensed scene and/or camera is in motion. In the second case, image lines are read one by one using moving mechanisms. There is however an offset between the end and the start of the exposure of consecutive lines such that an image line can start its exposure before the previous one has finished, or that each image line can start its exposure in a given frame just after finishing the previous frame, which maximizes frame rates [22]. Nevertheless, relative motion between the camera and scene can result in image shearing effects when a rolling shutter is used, which may require pre-processing in order to correct for this.

Depending on how many different channels in the electromagnetic spectrum (EM) that they are able to sense, cameras can be panchromatic (one wide band), RGB (three bands in the red, green and blue regions of the EM), multispectral (more than three bands across the EM) or hyperspectral (generally over 100 narrow bands across the EM). Thus, when an image is formed the data is distributed in a matrix of dimensions $m \times n \times p$, where $m \times n$ are the number of pixels organised in rows and columns and p the number of bands that the sensor is able to capture. The values within the matrix correspond to the intensity values acquired by each pixel in each sensed band.

2.2.2. Monocular cameras in space

Monocular cameras are frequently used for a variety of space applications. They are the most widely used sensor in Earth observation and are also used for relative navigation and planetary exploration.

There are ongoing efforts at both ESA and NASA to develop sensor suites for space rendezvous that would comprise both visual cameras and infrared cameras for situational awareness as well as LiDARs [24] [25]. An example of a successful rendezvous missions that used this technology for proximity operations is the manned Space Shuttle, where images taken by a camera helped as a visual aid to the crew for docking. There was also an image sensor called Proximity Sensor (PXS) on-board the ETS-VII flight experiment, developed by the National Space Development Agency of Japan (NASDA) as a precursor of the H-II Transfer Vehicle (HTV), launched in 1998. Commissioned by the U.S. Air Force Research Laboratory and launched in 2005, another example of a visual camera used for rendezvous and docking operations was on-board the mission XSS-11. This mission also included a LiDAR for relative navigation. Also on-board the NASA's Autonomous Space Transfer and Robotic Orbiter (ASTRO) is the Autonomous Rendezvous and Capture Sensor System (ARCSS), which includes an infrared visual camera and a laser range finder, among other sensors [26].

Figure 2.7 shows a stereo image pair acquired at our laboratory at Cranfield University. The images show a mock-up of the Envisat satellite.



Figure 2.7: Left and right undistorted and stereo rectified images acquired in our laboratory at Cranfield University. The images show a mock-up of the Envisat satellite.

2.3.2. Stereo cameras in space

Although stereo relative navigation for proximity operations is still in its early development stage and to the best knowledge of the author has not yet been implemented in space, it is currently being exhaustively researched. An example is the research done at ESA ESTEC in image processing for relative navigation for space applications [24]. In this work, the authors investigate stereo vision to obtain the depth of a scene that can be used for general relative navigation that can include space rendezvous, small body navigation and descent and landing for celestial bodies. Another research in this direction is that carried out as part of the Visual Estimation for Relative Tracking and Inspection of Generic Objects (VERTIGO) NASA's ISS-based research experiment by [28]. In their work, they tackle the particular problem of autonomous robotic vision-based circumnavigation manoeuvre about an unknown object based only on computer vision and inertial sensors. This work is in the frame of VERTIGO, that has the main aim of developing a solution for inspecting a non-cooperative target in space [28].

2.4. FPGA technology

2.4.1. Introduction to FPGA

Application Specific Integrated Circuits (ASICs) and Field Programmable Gate Array (FPGAs) are based on the same technology, i.e. both are built on silicon wafers with chemically diffused circuits and based on CMOS technology. The main difference however is that while ASICs are unique types of integrated circuits designed for a particular application, FPGAs are reprogrammable integrated circuits and are not limited to only one application [29]. This provides an advantage for FPGAs in that manufacturers can correct any error in the programming even once the FPGA is out of the production line, whereas this would be impossible with ASICs. For this reason, manufacturers often develop their prototypes on FPGAs so it can be tested and revised before producing the final product on an ASIC [30].

The main advantage of ASIC technology lies in their higher performance (without parallelization), with higher speed and lower power requirements than FPGA technology. Their main disadvantage is because they are application-specific, their manufacturing costs for low production and production times are higher than those of FPGAs [29].

2.4.2. FPGAs in space

When it comes to space applications, the main advantages of FPGAs are their re-programmability and structure that allows for parallelization, and so higher performance than an application-specific ASIC may be achieved. These characteristics also result in a final lower cost than for an ASIC, for low production.

Another advantage of reprogrammable FPGAs is that they can be reprogrammed in flight, which can be very useful for space missions that span many years [31].

However, an FPGAs' configuration is stored on SRAM memory, which makes them susceptible to single-event upsets (SEUs) due to space radiation, that can cause bit flips in configuration cells [31]. Because of this reason, even if they have been present in space for over a decade, their use has not been very successful within the European space sector. However, with the development of re-programmable FPGAs with over a million system gates and space-rated FPGAs that are resistant to radiation provided by companies such as Xilinx, their use in space has become more feasible [32].

Indeed, [33] highlighted the role that FPGAs can play for on-board processing in future NASA missions. As she stated, sensors on-board spacecrafts are becoming more advanced with a higher data rate that could reach up to 1–5 Terabytes per day of raw data. Being able to compress them on-board would significantly reduce the downlink requirements to download them. Furthermore, they would offer the possibility to upload on-board algorithms after launch. This could be achieved with space-qualified FPGAs such as the XilinxTM Virtex–4 and Virtex–5 series devices [33].

2.5. Conclusions

This Chapter introduced the sensors and hardware used in this research. In particular, the fundamentals of LiDAR sensors, monocular cameras and stereo cameras, as well as FPGA technology, have been discussed as well as their use in space.

Chapter 3 – Pre-processing algorithms

In order to improve the performance of the navigation algorithms, their input data must be optimised. This can be achieved by pre-processing the raw data acquired by the sensors (i.e. the high-resolution camera, the stereo camera and the LiDAR sensor). Pre-processing operations include spectral and spatial corrections for the visual images and outlier rejection and data down-sampling for the point clouds. Additionally, point cloud production through 3-D reconstruction from stereo images will be analysed. In this way, image-based data can be more easily fused with LiDAR data for a potential fusion-based navigation.

This Chapter is organised as follows: First, some camera notions necessary for the understanding of the image pre-processing techniques will be introduced. Then, image distortion sources and image processing operations performed in this work will be explained. Finally, common point cloud outlier sources as well as a method to reject them and an algorithm to reduce point cloud dimensionality will be described.

3.1. Preamble: Camera fundamentals

3.1.1. Image formation

The earliest prototype of the modern standard camera was the “camera obscura”, consisting in a dark room with a small hole or “pinhole” on one wall and a screen on the opposite wall. When the outside light enters the room through the pinhole, it casts an inverted image of the sunlit outside scene on the inner screen. By the end of 1600s, the hand-held camera obscura was already popular. Replacing the inner screen with a photosensitive surface, the camera as we know it today was born in 1826 [34].

This lens-less pinhole camera is the simplest camera model and, since its image formation only relies on a pinhole and not on focusing of the rays with lenses or mirrors, the aperture of this pinhole is what determines the quality of the image. In this way, pinhole apertures that are too high or too low result in blurry images. The pinhole aperture that produces the sharpest images is proportional to the distance to the sensed scene [34].

In the pinhole model, the outside scene can be projected on the camera plane using the perspective projection equations:

$$x' = \frac{f x}{z} \quad (3.1)$$

$$y' = \frac{f y}{z} \quad (3.2)$$

where (x', y') are the 2-D coordinates of the coordinates of a point on the image plane, (x, y, z) the 3-D world-coordinates of the point and f the camera focal length [35]. As shown in Figure 3.1, these equations can also be expressed in function of the effective focal length f' , i.e. the distance between the pinhole and the image plane, as [35]:

$$\frac{\mathbf{r}'}{f'} = \frac{\mathbf{r}}{f} \rightarrow \begin{cases} \frac{x'}{f'} = \frac{x}{f} \\ \frac{y'}{f'} = \frac{y}{f} \end{cases} \quad (3.3)$$

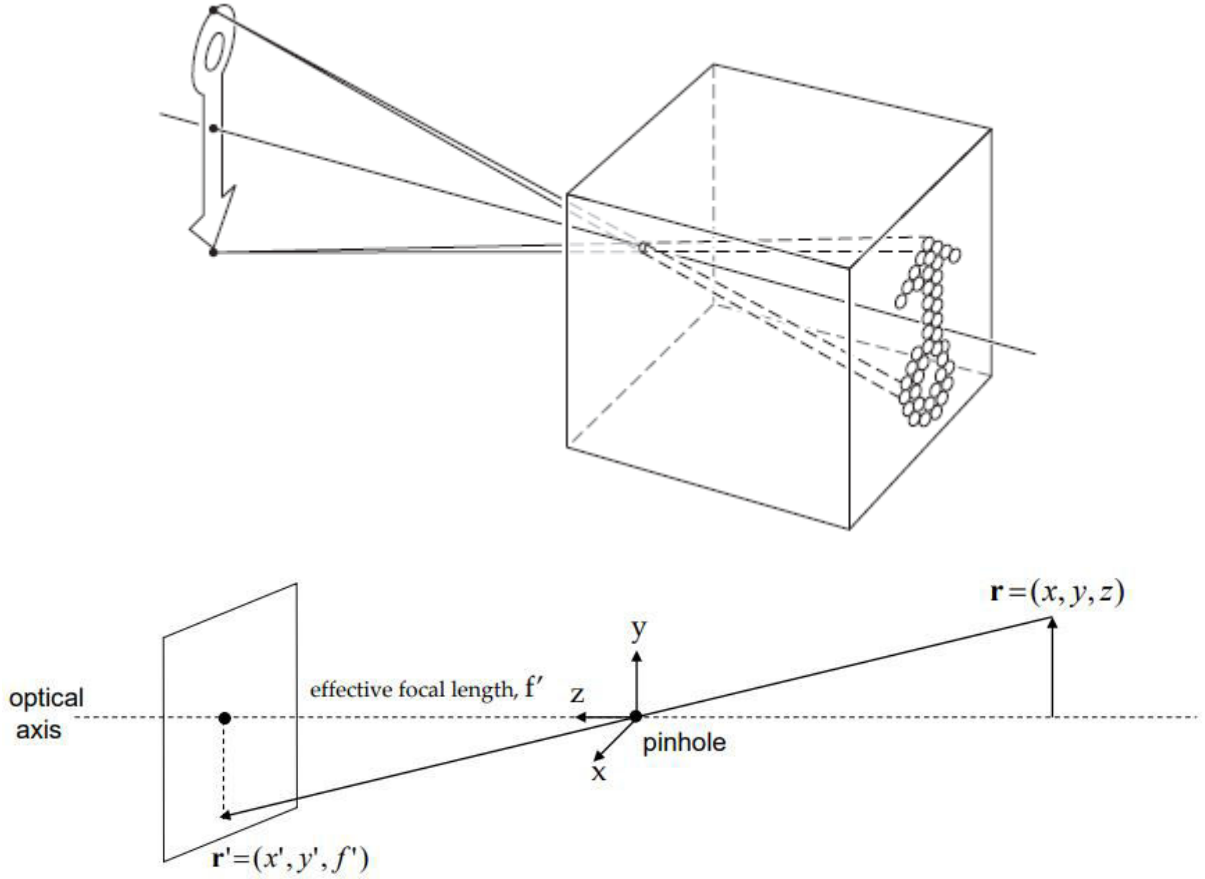


Figure 3.1: (Top) Pinhole camera schematic [34] and (bottom) geometric relationships [35].

By the addition of a lens to an optical system, it is possible to redirect the light rays emanating from the sensed objects, which leads to sharper images and an improved image quality. The “ideal lens”, called “thin lens”, is a lens that obeys the “Gaussian” or “first-order” optics laws. Basically, this means that the only light rays considered are in the paraxial region, this is, forming a small angle with the optical axis (small ϕ angle or when the points A and V are close in Figure 3.2). If ϕ is small, the relationships:

$$\cos \phi = 1 - \frac{\phi^2}{2!} + \frac{\phi^4}{4!} - \frac{\phi^6}{6!} + \dots \quad (3.4)$$

$$\sin \phi = \phi - \frac{\phi^3}{3!} + \frac{\phi^5}{5!} - \frac{\phi^7}{7!} + \dots \quad (3.5)$$

Can be approximated as $\cos \phi \approx 1$ and $\sin \phi \approx \phi$, so the only terms retained in the expressions are the first order terms. When this is not true, the resulting optics are “third-order” optics [34]. As far as image formation and processing is concerned, the main difference between first- and third-order

optics is that first-order optics obey a linear projection model in which straight lines in the scene result in straight lines on the image [36].

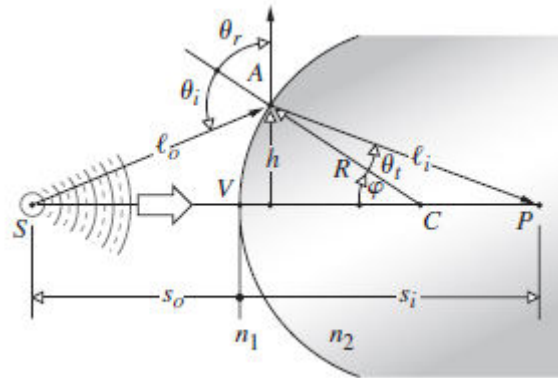


Figure 3.2: Refraction at a spherical interface [34].

A comprehensive mathematical explanation of Gaussian optics and thin lenses is presented in [34], pp. 163 – 168.

3.1.2. Camera parameters

The extrinsic and intrinsic parameters of a camera serve to locate the camera with respect to the outside world and to locate a pixel on the image with respect to the camera, respectively. Therefore, there are three coordinate systems that must be defined:

- *World frame*: fixed reference coordinate system for representing objects in the world. Distances are expressed in meters.
- *Camera frame*: reference coordinate system with the origin in the camera centre and the axis +Z along the camera optical axis, pointing to the exterior of the camera through the lens aperture. Distances are expressed in meters.
- *Image frame*: reference coordinate system to represent a point on the image plane. Distance are expressed in units of pixels.

The camera frame, although necessary to relate points from an image with the 3-D scene the image represents, is often unknown. The purpose of the **extrinsic parameters** is to locate the camera frame with respect to the world frame based on image information. They are a set of geometric parameters that uniquely identify the transformation between the unknown camera frame and the known world frame.

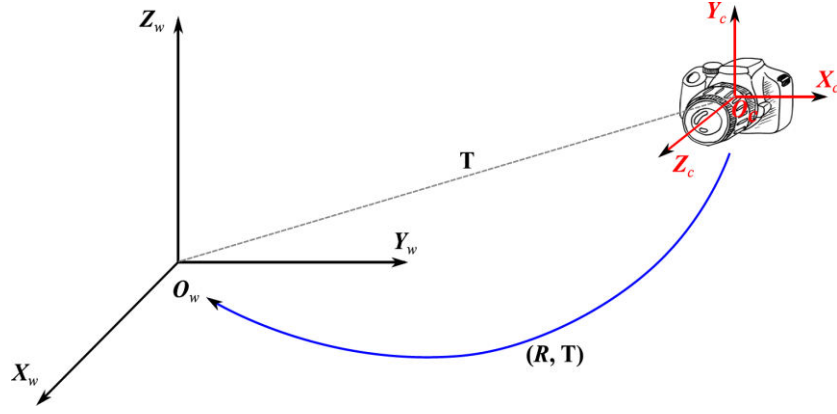


Figure 3.3: Conversion between the unknown camera frame and the known world reference frame [37].

Typically, these parameters take the form of a translation vector T (dimensions 3×1) and a rotation matrix R (dimensions 3×3). Therefore, the relationship between the coordinates of a point in the world frame $P_W = [X_W \ Y_W \ Z_W]^T$ and the same point expressed in the camera reference frame as $P_C = [X_C \ Y_C \ Z_C]^T$ can be expressed as [38]:

$$P_C = R(P_W - T_W) \quad (3.6)$$

or:

$$P_C = RP_W + T_C \quad (3.7)$$

Note that the two equations are different: the first equation represents a translation of the origin of the camera reference to the origin of the world reference frame expressed in the world reference frame (T_W) followed by a rotation from the world reference frame to the camera reference frame (change of coordinate system). The second equation represents a rotation from the world reference frame to the camera reference frame (change of coordinate system) followed by a translation of the origin of the world reference frame to the camera reference frame expressed in the camera reference frame (T_C). This transformation is depicted in Figure 3.3.

In total there are six extrinsic parameters of the camera, that correspond with the six degrees of freedom of a rigid body. They are represented in the three elements of the translation vector and the three independent elements of the rotation matrix.

The **intrinsic parameters** can be defined as the set of parameters needed to characterise the optical, geometric, and digital characteristics of the camera, allowing the pixel coordinates of an image point to be related to the coordinates of that point in the camera reference frame [38].

Image coordinates or “pixel coordinates” (X_s, Y_s) are expressed in units of pixels and have their origin c_s in the upper-left corner of the image plane, with positive axes x_s and y_s to the right and downwards, respectively, as shown in Figure 3.4. The pixel spacings are represented by s_x, s_y in the image.

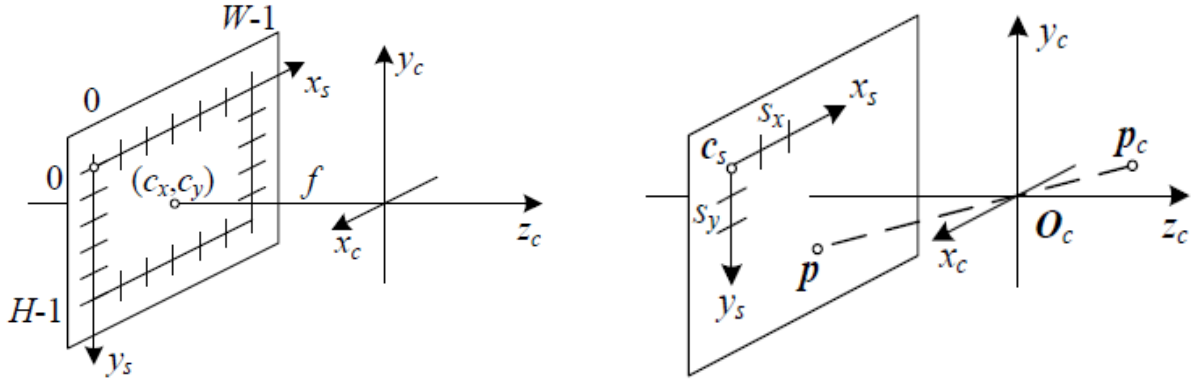


Figure 3.4: (Left) Camera frame and image frame showing the focal length f and the optical centre (c_x, c_y) . The image dimensions are $W \times H$. (Right) Projection of a camera-centred 3-D point p_c onto the image plane [36].

The transformation necessary to map 2-D pixel centres to 3-D rays is a sensor homography (i.e. isomorphism of projective spaces) M_s and can be expressed as [36]:

$$P = [R_s | c_s] \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_s \\ Y_s \\ 1 \end{bmatrix} = M_s \bar{x}_s \quad (3.8)$$

The resulting 3×3 M_s will be a matrix with the first two columns being 3-D vectors corresponding to unit steps on the image plane along x_s and y_s and with the third column being the 3-D coordinates of the image plane origin c_s .

The matrix M_s can be parametrized by eight unknowns: the two scale factors s_x and s_y , the translation c_s and the three independent elements of the rotation matrix R_s . Without precise external knowledge of the sensor spacing or sensor orientation and, there are only seven degrees of freedom instead of eight. This is because having only external image knowledge it is impossible to separate the translation c_s from the two scale factors [36].

The 3-D pixel centre P and the 3-D camera centred point P_c are correlated by an unknown scaling factor α as $P = \alpha P_c$. Therefore, the projection between P_c and a homogeneous version of the pixel address $\tilde{x}_s$ can be expressed as:

$$\tilde{x}_s = \alpha M_s^{-1} = K P_c \quad (3.9)$$

where K is the “calibration matrix” and represents the camera intrinsic parameters [36].

However, during calibration the intrinsic and extrinsic parameters of the camera are computed all at once as:

$$\tilde{x}_s = K [R | T] P_w = P P_w \quad (3.10)$$

where the 3×4 matrix P is called the “camera matrix” [36].

In order to resolve Equation (3.10) to extract the intrinsic and extrinsic camera parameters, it is necessary to reduce the number of degrees of freedom. This is done by triangularising the calibration matrix K so it is a function of a lower number of parameters. There are different possible

triangular forms of K [36]. The triangular form used by OpenCV and that will also be used in this work is:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (3.11)$$

with c_x and c_y being the coordinated of a principal point generally located at the image centre and f_x and f_y the independent focal lengths in the directions x and y of the sensor, expressed in pixel units [39].

3.2. Image processing

The proposed pre-processing pipeline for both the high-resolution and the stereo cameras within the I3DS project are depicted in Figures 3.5 and 3.6. In these block diagrams, the blue blocks represent standard image pre-processing algorithms (i.e. monocular), the black blocks correspond with the algorithms applied on pairs of stereo images and the yellow block represents point cloud pre-processing operations. The section in which each operation will be described is indicated in the respective block.

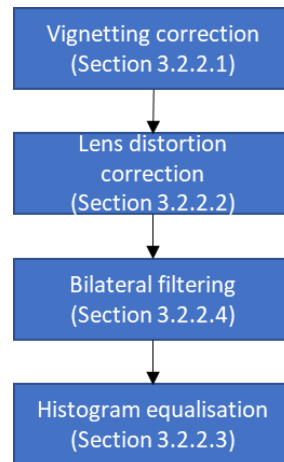


Figure 3.5: HR images pre-processing pipeline proposed within I3DS.

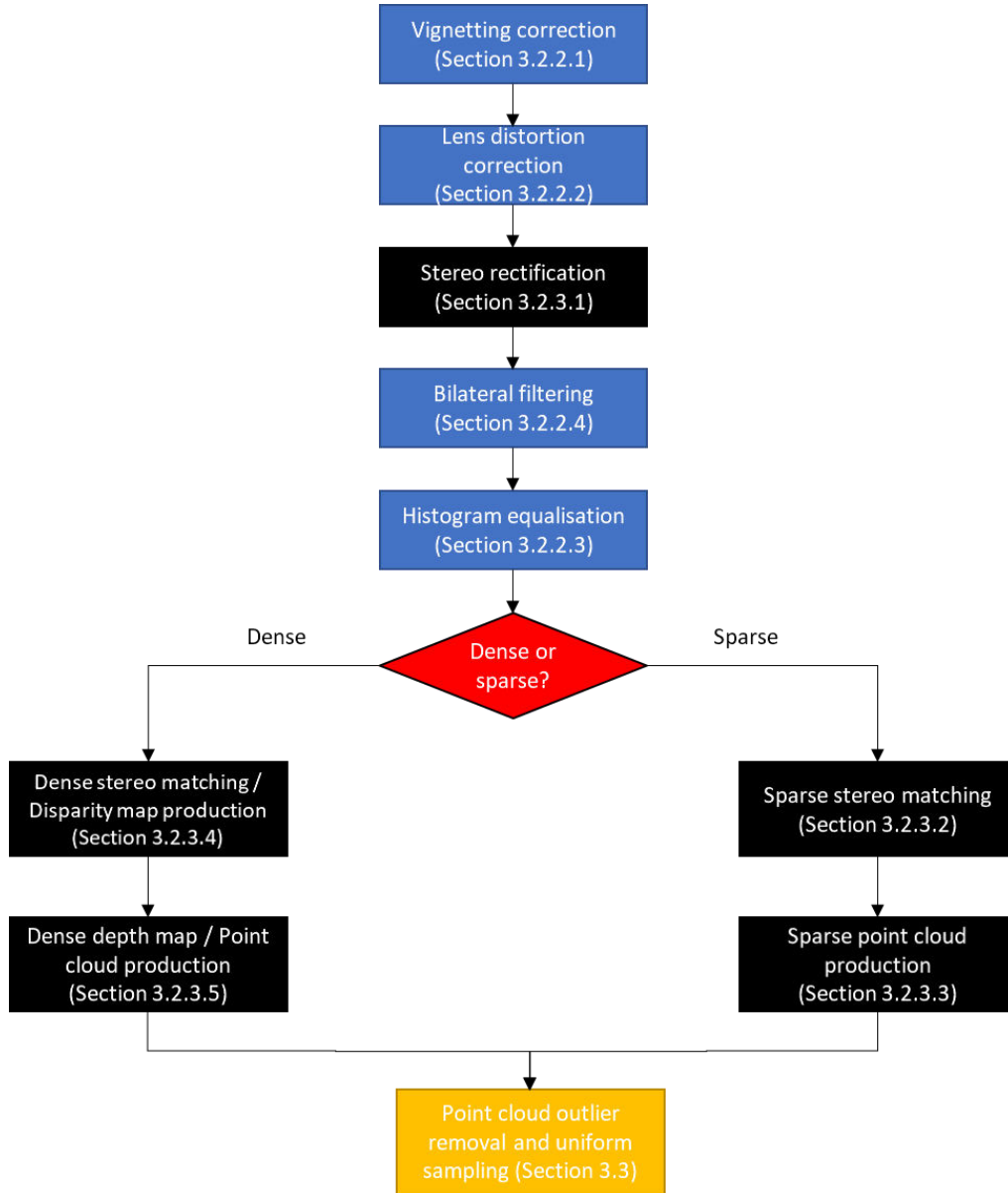


Figure 3.6: Stereo images pre-processing pipeline proposed within I3DS.

3.2.1. Sources of distortion in an image

The camera models previously described (pinhole and thin lens cameras) are “ideal”. In reality, lenses are typically spherical due to lower production costs and do not obey the first-order optic models. This results in non-ideal behaviour in the ray tracing and so in errors or “aberrations” in the image. There are two main kinds of aberrations, namely, chromatics aberrations and monochromatic aberrations. Chromatic aberrations are due to the fact that light and refraction characteristics depend on frequency or colour [34]. Finally, vignetting is another important common image distortion mechanism that results in a radial intensity fallout from the centre of the image. It can have different sources and is present in all images [40].

3.2.1.1. Seidel aberrations

Chromatic aberrations or Seidel aberrations are the direct consequence of the departure of the first-order assumption and were first studied by Ludwig von Seidel in the 1850s. They are schematised in Figure 3.7 and defined below:

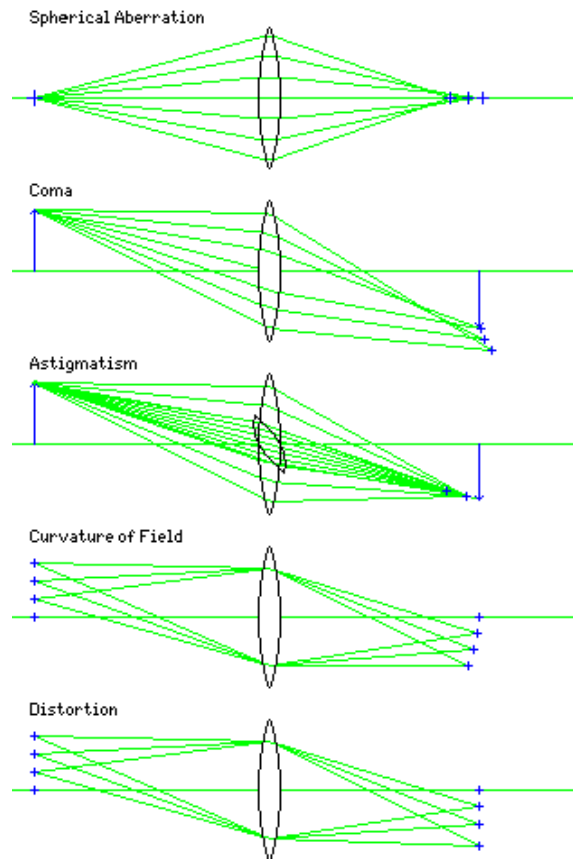


Figure 3.7: Diagram of the five Seidel aberrations [41].

- *Spherical Aberration*: spherical lenses and other non-ideal lenses can cause that rays coming from points on the optical axis and that pass through different parts of the lens are not focused at the same point at the other side of the lens [41].
- *Coma*: this aberration affects only to points off the optical axis but is similar to spherical aberration in that the lens focuses rays on the wrong places on the other side of the lens [41].
- *Astigmatism*: this aberration affects rays coming from points off the optical axis and that pass through a lens that is tilted from their point of view. They may not be affected by spherical aberration or coma, but because they pass through a part of the lens with a different profile, even if they come to focus this may not occur at the same distance from the lens [41].
- *Curvature of Field*: this aberration occurs when the light rays come to focus on a curved surface instead of a flat plane [41].
- *Distortion*: this is the most common Seidel aberration, because the other aberrations can be corrected with an adequate optics design. It occurs when scene points are casted on the image plane at a wrong distance from the optical axis [41]. There are three main types of distortion [36] [41] [42]:
 - Barrel distortion: straight lines in the scene curve outwards on the observed image or coordinates of points on the image are displaced away from the image center. It typically occurs in wide-angle lenses or at the wide end of a zoom lens.

- Pincushion distortion: straight lines in the scene curve inwards on the observed image or coordinates of points on the image are displaced towards the image center. It usually occurs in telephoto lenses, i.e. long-focus lenses in which the physical length of the lens is shorter than the focal length, or at the telephoto end of a zoom lens.
- Perspective distortion: parallel lines in the scene converge on the observed image. This is not really a distortion because it is a natural characteristic of 3-D vision. The same occurs in human vision but our mind processes the image to see the lines parallel. It occurs when the camera is not pointing at parallel lines perpendicularly.

Additionally, *fish-eye* can also be categorized as a type of distortion [36]. Barrel and pincushion distortions are radial distortions, while perspective distortion is simply a perspective problem, and fish-eye distortion is defined by more complex models than polynomial radial distortions [36].

3.2.1.2. Chromatic aberration

Due to the index of refraction for glass being dependent on wavelength and so on colour, standard lenses cause light of different colours to be focused at somewhat different distances and with a somewhat different magnification factor [36]. On the image, this aberration can be observed as colour fringes near high contrast edges [42]. Some manufacturers produce cameras with a specific optical design and different lenses in order to avoid this effect [36].

Chromatic aberration can be decomposed in three different phenomena [42]:

- *Lateral chromatic aberrations*: it takes the form of dual-colour fringing in an axial direction from the centre of the image and increasingly intensity towards the corners. It is the most easily correctable.
- *Axial chromatic aberrations*: appears as a single-colour halo around all high contrast edges. It is impossible to correct without having adverse consequences somewhere else in the image.
- *Sensor blooming*: particular to digital sensors, it causes spill-over highlight clipping and results in different colour fringing mostly around sharp highlights. It is usually impossible to correct.

3.2.1.3. Vignetting

To a greater or lesser extent, all images present an intensity fallout from the centre to the image or, in other words, the image darkens at the corners [36]. This is called vignetting and even when manufacturers try to design their lenses so as to minimise its effect, there is always some vignetting. It depends on aperture and focal length, in general increasing with the former and decreasing with the latter. This effect is more acute with wide-angle and wide-aperture lenses.

Vignetting can have many causes and can be classified according to different sources as:

- *Natural vignetting*: this type of vignetting consists of a radial intensity falloff due to the geometry of the optics and particularly depending on the incidence angle Θ with which a light ray enters the aperture of the camera. Firstly, when this angle increases with respect to the optical axis, the distance it must travel to the image plane increases, and thus the intensity of the light is reduced due to the inverse square law. Additionally, the camera aperture diameter is foreshortened for light rays arriving to the aperture at large incidence

angles, which leads to a reduction in light flux. This combined effect produces a radial intensity falloff from the centre of the image, more pronounced as the distance from the optical axis increased. This overall intensity falloff is often modelled as $\cos^4 \theta$ [40].

- *Pixel vignetting*: restricted to digital images, this kind of vignetting results from a reduction in sensitivity of the image sensor as the angle of incidence increases. This reduction in sensitivity occurs due to the finite depth of the photon wells in the digital sensor, which causes light striking a photon well at a steeper angle to be partially occluded by the sides of the well [40].
- *Optical vignetting*: this kind of vignetting, also called artificial vignetting or physical vignetting, is due to a physical blockage of the light rays by the lens diaphragm inside the lens body. Since small apertures limit light paths equally at the centre and edges of an image, optical vignetting can be reduced by limiting the aperture. It is thus dependent on the aperture width [40].
- *Mechanical vignetting*: this vignetting mechanism also is due to the physical blockage of light paths inside the camera. The difference with optical vignetting is that the light rays are not blocked by the lens but by other camera elements such as filters or hoods attached to the front of the lens body [40].

Natural vignetting, pixel vignetting and optical vignetting can be categorised under the term “internal vignetting” because they are due to the internal mechanisms of a particular camera and lens [42]. The two last vignetting mechanisms, i.e. optical vignetting and mechanical vignetting, are the easiest to correct. Optical vignetting can be corrected through a careful lens design of the optics in fixed focal length cameras or by limiting the aperture as mentioned above. On its side, mechanical vignetting can be easily eliminated when no external camera elements are being used. Thus, it is natural and pixel vignetting that require the most efforts to be suppressed or minimised.

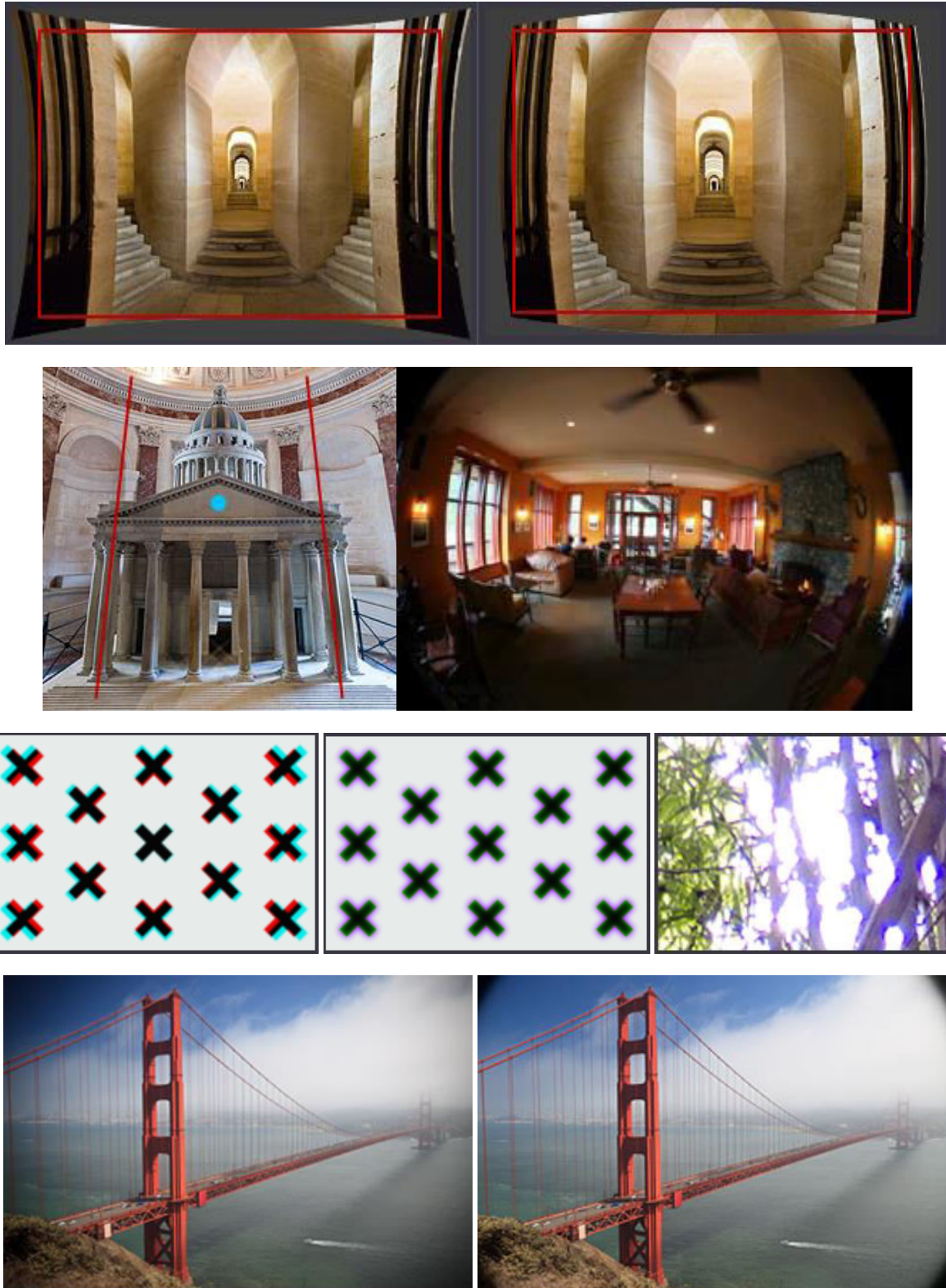


Figure 3.8: Image distortion mechanisms. From top to bottom and from left to right: Pincushion distortion [42], barrel distortion [42], perspective distortion [42], fish-eye [36], lateral chromatic aberration [42], axial chromatic aberration [42], sensor blooming [42], internal vignetting [42] and physical vignetting [42].

3.2.2. Image distortion correction methods

Although there are a wide variety of image pre-processing methods, this section will mainly focus on the methods that have been applied in this work.

3.2.2.1. Vignetting correction

Vignetting is known to be very detrimental for certain applications, including graphic applications, radiometric measurement applications and vision applications in which brightness constancy is essential to retrieve scene structure, such as stereo correlation and optical flow [40] [43]. This makes vignetting correction especially relevant for this work since the resulting pre-processed images will be input to the posterior navigation algorithms.

The most accurate way of calibrating a camera to correct vignetting is by using an integrating sphere or a picture of a uniformly coloured and illuminated blank wall [36]. Alternatively, Zheng, Lin and Kang [43] perform vignetting correction using only the information available from an image, by segmenting the image and applying information from reliable areas of the image to model the vignetting. Goldman and Chen [40] simultaneously determine the vignetting, exposure, and radiometric response function from a set of overlapping images, by assuming that vignetting is the same in each image and that the exiting radiance from a scene point in the direction of the camera is the same over all the images. In view that the most common vignetting correction methods in the literature assume radial vignetting, which is not always true and to avoid the need to have a calibrating image, Kordecki, Palus and Bal [44] developed a method based on the local parabolic model of non-radial vignetting and compensation of non-uniformity of scene luminance.

However, the approach of using calibrating images is always preferred if possible due to simplicity. In this work, because it was possible to calibrate the cameras using a blank uniformly illuminated wall, this was the method followed. All the pixels in such an image should have the same intensity values and so the vignetting effect can directly be observed. A radius-to-intensity attenuation mapping function can then be derived. This function will be used to correct posterior images acquired by the same camera by means of an intensity correction map involving only one addition or scaling operation for each pixel. This highlights the simplicity of this approach as it only involves one single operation per pixel. An exaggerated representation of vignetting over a blank wall as well as the resulting intensity correction maps and the corrected image for each case is shown in Figure 3.9.

Furthermore, in order to account for possible non-uniform illumination conditions, the cameras were rotated around their optical axes during calibration acquiring a set of images from different view angles. Then, the intensity value of each pixel was temporally averaged over the whole set of images. The resulting averaged image was then used to fit a 2-D polynomial function that approximates the radial intensity fall-off, by means of a least squares fitting algorithm. It is possible that this procedure could have introduced a discrepancy between the estimated radial vignetting function and the true vignetting if the true vignetting was not entirely radially symmetric since the procedure forced a certain amount of radial symmetry to exist in the temporally averaged image. However, the non-uniformity of the surface illumination was expected to be a more significant effect.

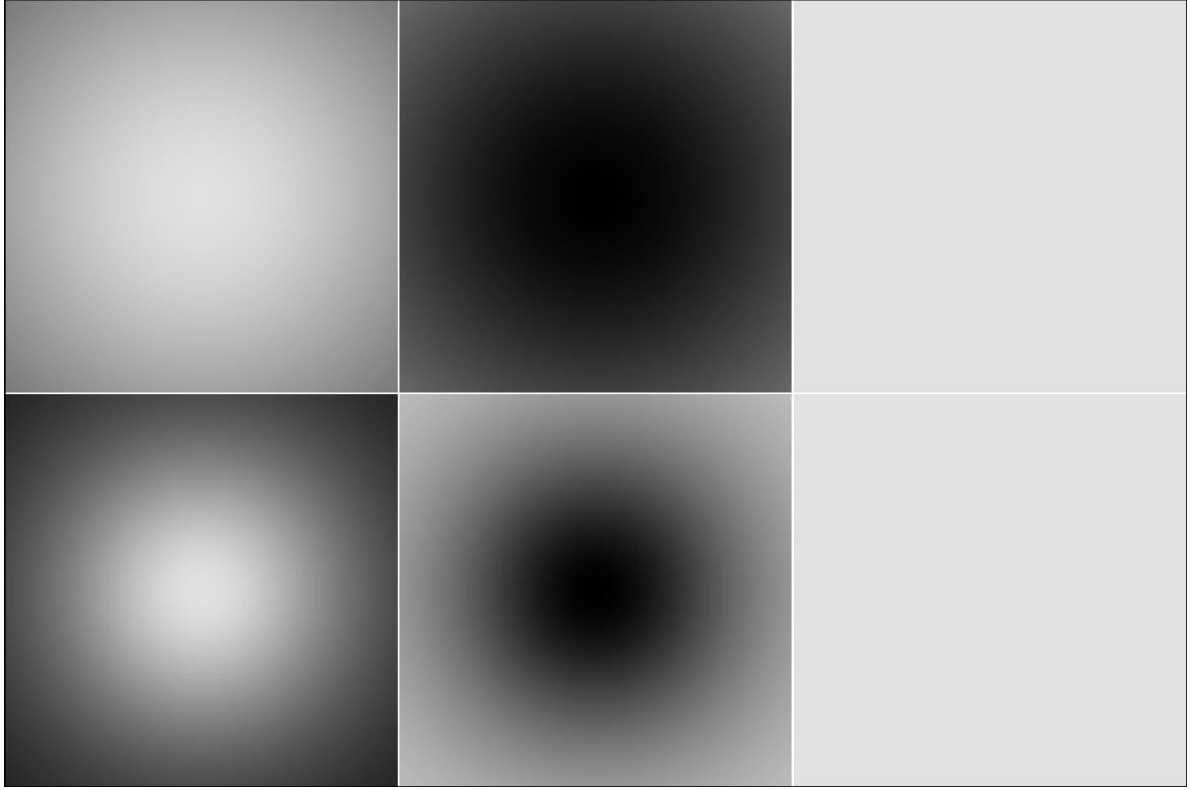


Figure 3.9: Exaggerated synthetic representations of vignetting over a uniformly illuminated blank wall (left), corresponding intensity correction maps (centre) and resulting corrected images (right).

3.2.2.2. Lens distortion correction

Once again, coherent and geometrically correct images are necessary in order to establish a correlation between them, either for stereo matching or for temporal matching between images for navigation purposes.

From Equations (3.8) to (3.11), it results:

$$\alpha \begin{bmatrix} X_s \\ Y_s \\ 1 \end{bmatrix} = K [R \mid T] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad (3.12)$$

where (X_s, Y_s) are the coordinates of the projection point in pixels and (X_w, Y_w, Z_w) are the coordinates of the scene point expressed in the world frame. Thus, we can re-write the model as [39]:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = R \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + T \quad (3.13)$$

$$x' = \frac{x_c}{z} \quad (3.14)$$

$$y' = \frac{y_c}{z} \quad (3.15)$$

$$X_s = f_x x' + c_x \quad (3.16)$$

$$Y_s = f_y y' + c_y \quad (3.17)$$

However, accounting for the radial and tangential distortion that real lenses have, their effect is modelled between Equations (14) and (15) and between Equations (16) and (17). The overall model is then extended to [39]:

$$x'' = x' \frac{1+k_1 r^2+k_2 r^4+k_3 r^6}{1+k_4 r^2+k_5 r^4+k_6 r^6} + 2 p_1 x' y' + p_2 (r^2 + 2 x'^2) \quad (3.18)$$

$$y'' = y' \frac{1+k_1 r^2+k_2 r^4+k_3 r^6}{1+k_4 r^2+k_5 r^4+k_6 r^6} + p_1 (r^2 + 2 y'^2) + 2 p_2 x' y' \quad (3.19)$$

$$\text{where } r^2 = x'^2 + y'^2 \quad (3.20)$$

k_1, k_2, k_3, k_4, k_5 and k_6 are the radial distortion coefficients and p_1 and p_2 the tangential distortion coefficients. Finally, the pixel coordinates can be written as [39]:

$$X_s = f_x x'' + c_x \quad (3.21)$$

$$Y_s = f_y y'' + c_y \quad (3.22)$$

In this work, geometric camera calibration is employed as a means of estimating the distortion coefficients, as well as estimating the focal length of the camera and the true location of the principal point. From these estimated parameters, an undistortion map can be pre-computed the same way that the intensity correction map was computed for vignetting correction. The undistortion map can be then used on each recorded image to correct for the distortions by remapping each image pixel to its correct location. This has the additional advantage of allowing the image formation process to be modelled by the simple geometry of the pin hole camera model, which greatly simplifies computer vision algorithms.

3.2.2.3. Histogram equalisation

Contrast enhancement algorithms amplify the intensity range in an image so that contrast between adjacent structures is maximally portrayed. Traditionally, these methods have applications in areas such as digital radiography where it is crucial to clearly differentiate between regions of interest [45].

Pixel intensity histograms represent the number of pixels or the relative frequency of occurrence of each intensity level that are present in an image. Histogram modelling methods process an image so that the histogram of the output image has the desired shape [45].

The histogram equalisation transformation, for an image of size $M \times N$, can be expressed as:

$$h(v) = \text{round} \left(\frac{cdf(v) - cdf_{min}}{(M \times N) - 1} \times (L - 1) \right) \quad (3.23)$$

where v represents a particular intensity value, $cdf(v)$ represents the cumulative distribution function, derived from the pixel intensity image, cdf_{min} is the minimum non-zero value of this function and L is number of grey levels used in the image (for example 256 for an 8-bit image).

In ordinary histogram equalisation, Global Histogram Equalisation (GHE) the image histogram is constructed from the entire image. This works well when the pixel intensities are similar across the

whole image, but in the existence of regions of high-contrast in the image, these regions will often not be sufficiently enhanced, producing a “washed-out effect”. Additionally, GHE tends to modify all intensities in the histogram towards left or right (i.e. dark or bright), which can lead to level saturation effects on some visibly important areas [46].

Adaptive histogram equalisation (AHE) can overcome this problem by dividing the image into tiles and for each pixel performing histogram equalisation using only its local neighbourhood or “contextual region” [47]. The intensity mapping function for AHE can be expressed as [47]:

$$m(i) = Display_range * \frac{Cumulative_Histogram(i)}{Region_size} \quad (3.24)$$

However, adaptive histogram equalisation can result in over-amplification of the image noise in flat regions in the image and “ring” artefacts at strong edges. This can be avoided using Contrast Limited Adaptive Histogram Equalisation (CLAHE), a technique with a higher flexibility in choosing the local histogram mapping function. Basically, the clipping level of the histogram can be selected, which can minimise the noise amplification produced by AHE [45].

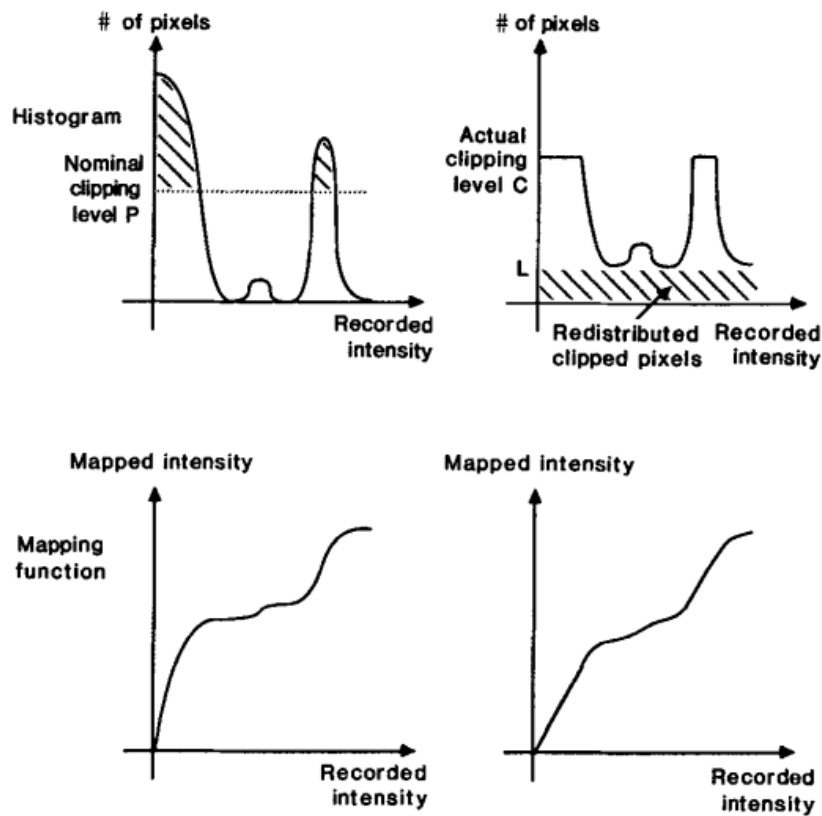


Figure 3.10: Contrast mapping functions and their associated original and clipped histograms [47].

Contrast enhancement can be understood as the slope of the intensity mapping function in Figure 3.10: a slope of 1 means no enhancement and larger slopes represent larger enhancements. The derivation of the intensity mapping function and therefore its slope results:

$$\frac{dm}{di} = \frac{Display_range}{Region_size} * Histogram(i) \quad (3.25)$$

that is proportional to the histogram. Thus, the contrast enhancement is proportional to the histogram at each level of intensity. Therefore, clipping the slope of the mapping function corresponds to clipping the height of the histogram. Clipping the height of the histogram can be done by limiting the pixel count at each intensity level and re-distribute the spare pixels (i.e. over the clipping number) over the rest of intensities. This is necessary to ensure that the whole input intensity range is mapped to the output intensity range. After evaluating different options, Pizer et al. [47] concluded that the most appropriate method to redistribute the spare pixels was to do a uniform distribution over the whole intensity range in the whole image, due to simplicity and also to avoid undesired large intensity modifications in the image. This operation is implemented in the AHE algorithm by modifying the histogram for each contextual region once it is computed and before applying the mapping function [47].

3.2.2.4. Bilateral filtering

Bilateral filtering is a smoothing, edge-preserving algorithm. It substitutes the value of each pixel by a non-linear combination of intensity values of the neighbour pixels, which removes image noise. It is a non-iterative, local and simple method. It combines intensity values based on their geometric nearness and photometric similarity.

The difference from a standard Gaussian noise-reduction algorithm is that it not only takes spatial information into consideration, but also intensity information. Low-pass Gaussian filters assume that intensity varies slowly over space and so nearby pixels will have a higher similarity in intensity values. However, this hypothesis is not true for edges, that result in being blurred after applying Gaussian filters. On the contrary, bilateral filtering also takes photometric closeness into consideration. In this way, neighbouring intensities are given a higher weight than more distant intensities, allowing the more distant intensities to be preserved. Ultimately, this allows the preservation of edges [48].

Tomasi and Manduchi [48] developed bilateral filtering by combining two different kind of filters, differenced by how the intensity values are averaged. On the one hand, domain filtering assigns a higher weight to spatially close pixels. On the other hand, range filters assign a higher weight to pixels that are similar in intensity. Although, contrary to domain filters, range filters preserve colours, they also distort an image's colour map when used in isolation. Bilateral filtering is the combination of both. It is thus non-linear because range filters assign different weights based on intensity and colour [48].

Mathematically, domain filtering applied to an image $f(x)$ can be expressed as:

$$h_d(x) = k_d^{-1}(x) \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\varepsilon) c(\varepsilon, x) d\varepsilon \quad (3.26)$$

where $c(\varepsilon, x)$ represents the spatial closeness between the neighbourhood centre x and a nearby point ε . The analytical expression of the normalization constant, $k_d(x)$, is [48]:

$$k_d(x) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} c(\varepsilon, x) d\varepsilon \quad (3.27)$$

that is constant when the filter is shift invariance, since in that case $c(\varepsilon, x)$ is only a function of the difference, $\varepsilon - x$.

On its side, range filtering can be written as [48]:

$$h_r(x) = k_r^{-1}(x) \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\varepsilon) s(f(\varepsilon), f(x)) d\varepsilon \quad (3.28)$$

where $s(f(\varepsilon), f(x))$ represents the photometric similarity between the centre x and the point ε . In this case, the normalization constant k_r is:

$$k_r(x) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} s(f(\varepsilon), f(x)) d\varepsilon \quad (3.29)$$

constant when the similarity function s is “unbiased”, i.e. only depends on the difference $f(\varepsilon) - f(x)$.

Finally, the combination of both, bilateral filtering results [48]:

$$h(x) = k^{-1}(x) \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\varepsilon) c(\varepsilon, x) s(f(\varepsilon), f(x)) d\varepsilon \quad (3.30)$$

with:

$$k(x) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} s(f(\varepsilon), f(x)) d\varepsilon \quad (3.31)$$

3.2.3. Other image processing algorithms: Stereo images

As can be expected, the pre-processing algorithms performed on individual stereo images are the same as those performed on monocular images. However, in order to acquire the depth information that stereo vision can provide, it is necessary to carry out some additional processing operations on each pair of stereo images. Firstly, it is necessary to correlate the points or features of the sensed scene that are visible in both images. Once these common points are found, their 3-D coordinates can be found through triangulation. Finally, a point cloud can be built from the 3-D coordinates of a set of points, if needed. For example, data in a point cloud format can be integrated with LiDAR data to achieve a posterior combined navigation solution.

Before proceeding with the explanation of the methods, it is necessary to define the basic concepts of epipolar geometry. Having a stereo setup, the **epipole** is the intersection point of the line joining the two optical centres (i.e. baseline) with the image plane. It can be seen as the visualization of the optical centre of one of the cameras on the image plane of the other. The **epipolar plane** is the plane defined by the two optical centres and a 3-D point in the sensed scene, or an image point, and the two optical centres. Finally, the **epipolar line** is the straight line of intersection of the epipolar plane with the image plane [49].

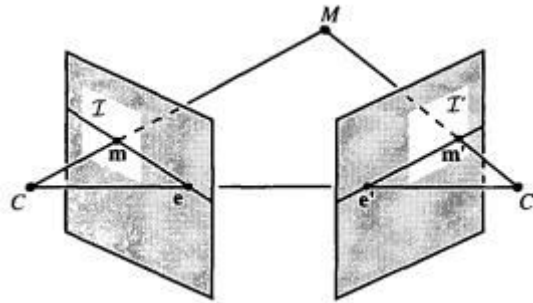


Figure 3.11: Epipolar geometry between a pair of images [50].

3.2.3.1. Stereo rectification

Ideally, two cameras in a stereo setup should have their optical axes perfectly aligned and coplanar image planes. However, due to manufacturing errors and imperfections in the cameras optical

systems, this is extremely unlikely. In this situation, the application of stereo geometry to obtain the scene depth is not a straight-forward process.

The goal of stereo rectification is to transform a pair of stereo images with known epipolar geometry into two output images so that epipolar lines in the original images are horizontally aligned in the transformed images. This is done by applying a pair of homographies to the two original images [50].

In this way, corresponding features on the two rectified images will be on the same horizontal scan line and only differ in horizontal displacement, i.e. disparity. Thus, corresponding pixels will have the same y image coordinate and only vary in the x coordinate. Furthermore, since disparity is due to the scene depth, it can serve to retrieve it. Figure 3.12 shows a pair of stereo rectified images. As can be seen from the horizontal lines matching corresponding features, each pair of corresponding features has the same y -coordinate on the image.



Figure 3.12: Pair of stereo rectified images. Horizontal lines match corresponding features [39].

As seen in Section 3.1.2, the relative pose (translation and rotation) of a camera with reference to a known world frame can be obtained in form of the camera extrinsic parameters through calibration. Many stereo rectification algorithms ([51], [52], [53]) use calibration information together with stereo geometry to transform a pair of stereo images to obtain a pair of output images like those that would be produced by a rectilinear stereo rig (i.e. a stereo setup with coplanar image planes and parallel optical axes).

However, calibration may not always be a possibility. In view of this, Loop and Zhang [50] developed a rectification method using epipolar geometry and that is based on the 2-D image information. The method is valid for calibrated and uncalibrated cameras, as far as the fundamental matrix F (i.e. a 3×3 matrix that maps points in one image to the corresponding epipolar line in the other image) is known.

The method used in this work is the implemented by OpenCV [39] and corresponds with the algorithm developed by Hartley [54], that performs stereo rectification based on 2-D projective transformations. As the method developed by Loop and Zhang [50], it does not need previous calibration and is purely based on the 2-D images. The main difference with that method is that it does not need the fundamental matrix F to be known beforehand but computes it instead.

The input to the algorithm are a pair of stereo images with a common overlap region and the output, a pair of stereo rectified images so that the epipolar lines in the two images are horizontal. It consists of the following steps [54]:

1. Identify a set of at least seven image-to-image matching points $u_i \leftrightarrow u_i'$ between the two images.
2. Calculate the fundamental matrix F and the epipoles p and p' in the two images. The fundamental matrix F can be expressed as:

$$u_i'^T F = u_i^T \quad (3.32)$$

thus, having eight corresponding points the fundamental matrix can be extracted as the least-squares solution for Equation (3.32).

3. Compute a projective transformation H' that maps p' to the point at the infinity, $(1,0,0)^T$. The reason behind this operation is that in order to map the input images to a rectilinear stereo rig, the epipolar lines must be transformed to lines parallel to the x-axis in the image. As explained in [54], a projective transformation H that makes the epipole parallel to the x-axis has four degrees of freedom, and a wrong conversion can induce important geometrical distortions in the image. However, imposing that the transformation must be that of a rigid body (i.e. defined by a translation and a rotation) around the centre of the image u_0 , the resulting transformation is:

$$H = G R T \quad (3.33)$$

where T is a translation vector taking the point u_0 to the origin, R is a rotation matrix that takes the epipole p' to a point $(f, 0, 1)^T$ on the x-axis and G is a transformation that takes the epipole $p = (f, 0, 1)^T$, that lies on the x-axis, to the point in infinity $(f, 0, 0)^T$. The analytical expression of G is:

$$G = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{1}{f} & 0 & 1 \end{pmatrix} \quad (3.34)$$

that approximates to the identity mapping in the origin.

4. Find the projective transformation H that minimizes the least-squares distance:

$$\sum_i d(H u_i \cdot H' u_i') \quad (3.35)$$

5. Resample the first image using the projective transformation H and the second image using the projective transformation H' .

3.2.3.2. Sparse stereo matching

Stereo matching consists of finding the points in a pair of stereo images that correspond to the same 3-D point in the scene. As explained in Section 3.2.3.1, once the stereo images have been rectified corresponding points lie on the same row in both images and so this search is limited to the horizontal dimension of the images.

Some computer vision applications, including many data fusion algorithms for navigation purposes, are best suited to sparse observations of highly discriminatory image features. In these cases, a

collection of keypoints must be first identified and described in one or both images of the stereo pair using a feature detection and description method.

Among the many feature detection and description methods found in the literature, the most popular ones, in order of appearance, are SIFT (2004) [55], SURF (2006) [56] and ORB (2011) [57]. Although SIFT has proved successful in a number of applications that use visual features, such as object recognition or image stitching, it has the main drawback of a high computational load that results in high computing times. In view of this, there has been great research efforts to reduce this time. Out of the algorithms developed, the most widely used and with the best performance has been SURF. However, the accuracy it achieves is lower than that of SIFT [57]. In 2011, Rublee et al. [57] developed the ORB algorithm, that combines the FAST keypoint detector [58] and the BRIEF descriptor [59].

Karami, Prasad and Shehata [60] compared these three methods (i.e. SIFT, SURF and ORB) for a series of transformations and image distortions, namely, scaling, rotation, noise, fisheye distortion, and shearing. They found ORB to be the fastest and SIFT to outperform the other two algorithms in almost every situation. An exception was rotations proportional to 90 degrees, where SURF and ORB outperformed SIFT and in images with a high level of noise ORB and SIFT performed very similarly. Finally, while SIFT and SURF found keypoints over a whole image, ORB found them mostly in the centre of the image [60].

Computationally, the feature matching step can be done with the FLAAN (Fast Library for Approximate Nearest Neighbors) libraries. These libraries are developed in C++ and have bindings for C, Matlab and Python. They contain algorithms to perform ‘fast approximate nearest neighbour searches in high dimensional spaces’ and a system to automatically choose the best algorithm and parameters depending on the dataset [61] [62].

However, the sparse stereo matching method used in this work is the the Kanade-Lucas-Tomasi feature detection and tracking algorithm (KLT). The concept of the KLT feature tracker first appeared in Lucas and Kanade’s paper [63] in 1981 and was fully developed by Tomasi and Kanade [64] in 1991. It was further extended in 1994, when Shi and Tomasi [65] presented a KLT-based method to select “good features” to track. The choice of the KLT algorithm over the SIFT, SURF and ORB is based on a lower computational load and computing time. Because this algorithm is expected to run on an FPGA board, computational load is a determinant factor.

Specifically, this method has the following phases in the present work:

1. A KLT feature detector is used on the left-camera image. The selection of left-camera image as reference frame for the pose estimation does not lead to a lack of generality, providing all successive computation steps are relative to it.
2. The location of detected features is refined to sub-pixel precision. It is worth noting that the sub-pixel accuracy refinement is done in a window around the feature location. Hence, the size of the window can affect the process if not properly tuned for the entire range of distances envisaged in the operations.
3. The sub-pixel refined positions of each detected feature in the left-camera image and the entire right-camera image are fed into a KLT tracker to obtain stereo matching of features. That is, given the location of feature in the left-camera image and a given window size, the tracker searches in the right-camera image for the corresponding point, whose surroundings

within the window maximize the probability of a match. Even in the case of stereo matching with the KLT tracker, the window size has great importance, since it has to be sufficient to cover the entire range of disparities that can be encountered in the space mission profile.

4. A preliminary outlier removal is performed and pairs of features that do not respect the stereo constraint or that do not lie in the same epipolar line are discarded.

3.2.3.3. Sparse stereo point cloud production

Once the pairs of corresponding features are found in both images, triangulation (see Section 2.3.1) can be applied to compute the 3-D coordinates of each feature in the scene. As seen in Chapter 2, triangulation is the problem of, having the image coordinates of the projection of a scene point in both images of a stereo pair, finding the intersection of the two rays in space. This intersection will correspond with the scene point location in space.

Ideally, i.e. in the absence of noise, this operation would be straight-forward through the triangulation equation (see Section 2.3.1). However, noise and imperfect systems lead to the two rays potentially not crossing and thus, an optimum solution must be found.

There are many triangulation methods that include non-iterative and iterative linear methods, a “mid-point method” based on geometry, and polynomial method [66].

In this work a non-iterative linear least-squares triangulation method [66]. Linear triangulation makes use of the relationships expressed in the Equations (3.10) and (3.12) to establish:

$$\alpha \begin{bmatrix} X_S \\ Y_S \\ 1 \end{bmatrix} = P X \quad (36)$$

where the camera matrix P is known from calibration and the pixel coordinates of a feature in the image (X_S, Y_S) are also known. The unknowns are then the scale factor α and the world coordinates of the scene point X . Calling P_1, P_2 and P_3 the vectors forming the three rows of the camera matrix P , the equation can be decomposed in the following three:

$$\alpha X_S = P_1^T X \quad (3.37)$$

$$\alpha Y_S = P_2^T X \quad (3.38)$$

$$\alpha = P_3^T X \quad (3.39)$$

from where substituting the equation for the scale factor in the other two and re-ordering, it results:

$$(X_S P_3^T - P_1^T) X = 0 \quad (3.40)$$

$$(Y_S P_3^T - P_2^T) X = 0 \quad (3.41)$$

Note that these equations have been developed only for the pixel coordinates of the scene point in one of the cameras. The development for the other camera would be similar, with different pixel coordinates (the corresponding x_s') and a different camera matrix P^2 , also known from calibration. Thus, it would result:

$$\begin{pmatrix} X_S P_3^{1T} - P_1^{1T} \\ Y_S P_3^{1T} - P_2^{1T} \\ X'_S P_3^{2T} - P_1^{2T} \\ Y'_S P_3^{2T} - P_2^{2T} \end{pmatrix} X = 0 \quad (3.42)$$

That can be treated as a system of the form $Ax = 0$, with A being a matrix of dimensions 4×4 . Finally, the least-squares method [67] reduces the vector X to $X = (X_w, Y_w, Z_w, 1)^T$ and thus, the system is reduced to a set of four equations with three unknowns, namely, X_w , Y_w and Z_w . In this work, this system is solved using singular value decomposition (SVD) [68].

After calculating the 3-D coordinates of the scene points, the data can be represented as a 3-D point cloud for posterior navigation and integration with data from other sources such as LiDAR.

3.2.3.4. Dense stereo matching / disparity map production

On the contrary to sparse stereo matching, that establishes only correspondences of a set of features or keypoints in a stereo pair, dense stereo matching aims to find correspondences for the whole image. These methods can generally be categorised into local methods, pixelwise matches and global approaches [69].

The cost function to assess the performance of the different methods is computed based on the intensity difference between matches. However, intensity is susceptible to recording and illumination differences between the two cameras. Mutual Information (MI) [70] appeared as a solution to this problem, as an image matching method robust to complex intensity relationships and reflections [69].

Semi-Global Matching (SGM) has been established as one of the most widely used dense stereo matching algorithms since it first appeared in [69] and is the method used in this work. The problem is formulated having a pixel p in one image, with intensity I_{bp} , and a suspected correspondence $q = e_{bm}(p, d)$ with intensity I_{mq} , where the function $e_{bm}(p, d)$ represents the epipolar line in the match image for the base image pixel p with the line parameter d . For rectified images, $e_{bm}(p, d) = [p_x - d, p_y]^T$, where d is the disparity [69].

Firstly, the matching cost is calculated pixelwise based on MI from the entropy H of the two images as well their joined entropy. Entropy in this context refers to the “information content” of an image and is calculated from its probability distribution P of intensities. The joined entropy considers the probability distributions of the two images. The basic form of the MI and the involved entropies are calculated as:

$$MI_{I_1, I_2} = H_{I_1} + H_{I_2} - H_{I_1, I_2} \quad (3.43)$$

$$H_I = - \int_0^1 P_I(i) \log P_I(i) di \quad (3.44)$$

$$H_{I_1, I_2} = - \int_0^1 \int_0^1 P_{I_1, I_2}(i_1, i_2) \log P_{I_1, I_2}(i_1, i_2) di_1 di_2 \quad (3.45)$$

For a further development of this model and the extraction of the MI matching cost refer to [69].

Because pixelwise cost computation can have errors due to noise, an additional constraint that penalises neighbouring disparities is added as a smoothing step. This is calculated at the

“aggregation cost” and expressed in the following equation, where the energy $E(D)$ depends on the disparity image D .

$$E(D) = \sum_p \left(C(p, d) + \sum_{q \in N_p} P_1 T[|D_p - D_q| = 1] + \sum_{q \in N_p} P_2 T[|D_p - D_q| > 1] \right) \quad (3.46)$$

Where the first term is the sum over all the pixel of the matching cost, the second term adds a penalty P_1 the neighbouring pixels that differs in disparity from the original pixel and the third term adds a larger penalty P_2 for larger disparity [69]. In this way, the problem of stereo matching is reduced then to calculate the disparity image D that minimizes the $E(D)$.

3.2.3.5. Dense depth map / point cloud production

Once the disparity map has been computed, the data can be transformed into a 3-D point cloud through triangulation as was done for sparse point cloud production in Section 3.2.3.3.

3.3. Point cloud pre-processing

Either from the LiDAR sensor or derived from stereo images, point clouds need to be pre-processed before being passed as input to the posterior navigation algorithms. Point clouds must be first cleaned from possible spurious features, through an outlier removal filter, and then downsized in order to make them more easily handled posteriorly.

3.3.1. Sources of distortion

Hawkins [71], as cited by [72], defined an outlier as “an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism”.

Image-based 3-D reconstruction methods are prone to producing outliers and noise in the depth maps because of matching ambiguities and image imperfections (lens distortion, sensor noise, etc.). Therefore, point clouds obtained from stereo images are generally noisier than those acquired actively such as by laser scanning [73].

According to Sotoodeh [72], the main mechanisms that cause outliers in laser scans are:

- *Boundaries of occlusions*: when a laser beam hits the border of an occlusion (i.e. an object occluding other objects in the scene) its footprint is divided into two parts. One of these parts of the beam radiates on the “front” and the other one on the “rear” surfaces incident to the occlusion boundary. Therefore, the signal received at this point is a weighted average of the irradiance reflected by both surfaces. This leads to a virtual point to be created between the front and the rear surfaces and while the beam moves across the occlusion boundary, artificial points are created between the two surfaces (see Figure 3.13) [72].
- *Surface reflectance*: laser scanning acquires good measurements on surfaces that are opaque and diffusely reflecting. However, translucent and non-homogeneous materials such as glass or plastic can reflect a beam that the detector is not able to resolve. This can cause two main measurement errors: a wrong measured distance and a higher level of noise [72] [74].
- *Multi path reflection*: when the LiDAR laser beam reaches a surface with a very low incidence angle, it can reflect in a specular fashion and send the pulse in some direction other than the sensor direction before being reflected back from other surface and into the

sensor. This secondary reflection, more effective than the first one, will result in an outlier [72].

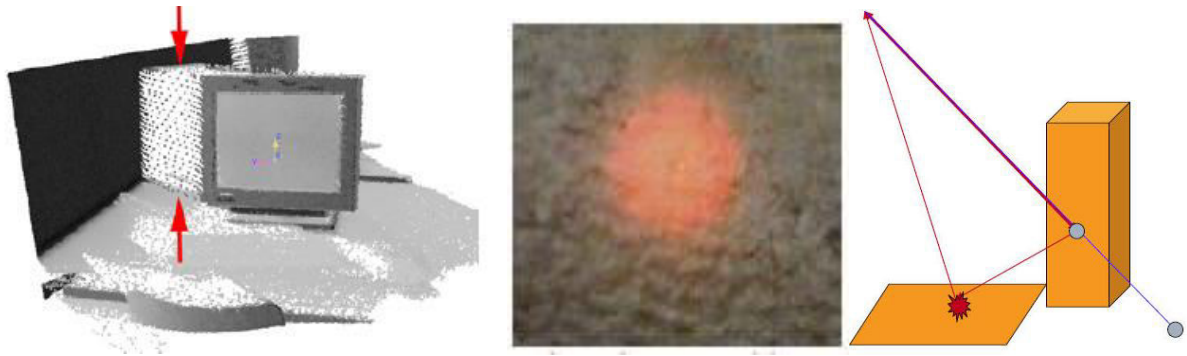


Figure 3.13: LiDAR error sources. From left to right: reflection on the boundary of a occlusion (desktop computer) has created outliers between the computer and the rear wall [72]; laser reflection on a translucent and non-homogeneous marble surface [74]; multipath reflection causes the sensor to compute a point that was never measured [75].

3.3.2. Outlier removal

Outlier removal has the ambition to clean the 3-D point cloud data from spurious features that may negatively impact the performance of the navigation algorithms.

Papadimitriou et al. [76] classified general outlier rejection algorithms for general datasets into the following categories:

- *Distribution-based approach*: It assumes that objects in the dataset conform to a certain standard statistic distribution and identifies as outliers those objects that deviate from this distribution.
- *Depth-based approach*: Based on computational geometry, this approach divides the data in the dataset into different layers of k -d convex hulls. Objects in the outer layer are flagged as outliers.
- *Clustering approach*: In general, the main goal of clustering algorithms is not outlier rejection and outliers are detected as an additional consequence of data clustering itself. Therefore, outlier rejection criteria are not explicit.
- *Distance-based approach*: The general statement of these methods says that “an object in a dataset P is an outlier if at least a fraction β of the objects in P are further than r from it”.
- *Density-based approach*: This approach assigns to each point a “local outlier factor” (LOF) (i.e. the “degree of being an outlier”) depending on the local density of its neighbourhood. Specifically, LOF represents how isolated an object is from its neighbourhood. Thus, objects with a high LOF are classified as outliers [77].

For image-based point clouds, there are some outlier rejection methods that rely on image information. For example, Wolff et al. [73] developed a method that compares each point with the surface implied by its surrounding points, from both a geometric and photometric point of view. Inconsistent points are then interpreted as outliers and removed from the point cloud. However, photometric comparison cannot be done for LiDAR-based point clouds.

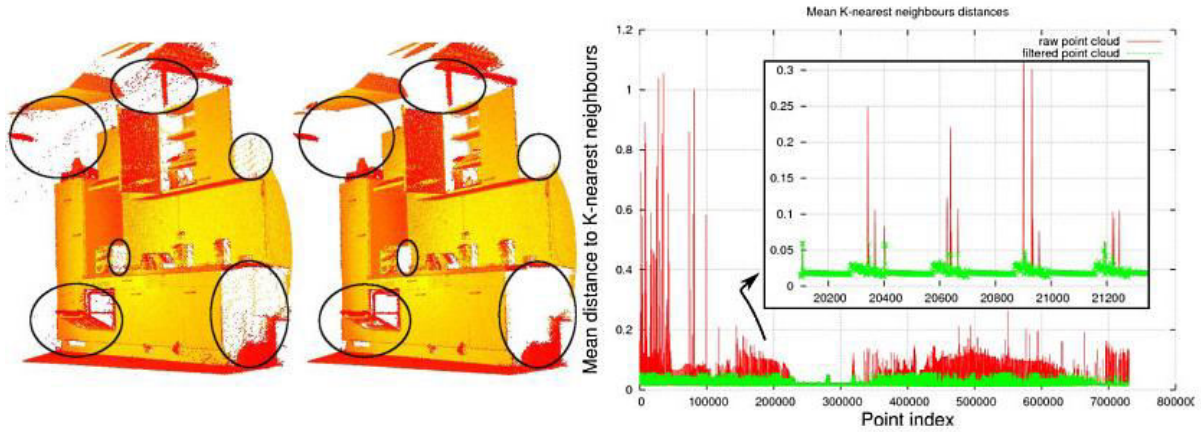


Figure 3.14: Result of applying the PCL statistical outlier removal to a point cloud. On the left, the input and output point clouds are represented. The graph on the right shows the mean k -nearest neighbour distances in a point neighbourhood before and after filtering [78].

Thus, the outlier rejection filter applied in this work is a statistical outlier rejection method that only relies on the geometric properties of the point clouds. In general terms statistical outlier removal consists in performing a statistical analysis in a point's neighbourhood and discarding the points that do not meet a certain criterion. The specific method used in this work is the developed by the open Point Cloud Library (PCL) and is based on the mean distance of the points to their neighbours. Assuming that the distribution of these distances is Gaussian with a mean μ and a standard deviation σ , all the points that do not conform to this distribution (i.e. their mean distance are outside the interval defined by the global distances mean and standard deviation) are filtered out of the point cloud [78].

3.3.3. Point cloud down-sampling: Uniform sampling

Point clouds can have up to several thousand points and can be very heavy. For example, as will be seen later, the raw point clouds used in this work were in the order of 5 MB (with 152 100 points), whereas the raw images representing the same scene were in the order of 0.5 MB. This, together with the complexity of point cloud operations that treat all points in the cloud, results in high computing requirements. Because the intention of this work is to develop a real-time navigation solution, it is necessary to reduce the size of the point clouds to reduce the necessary computing time per frame.

The down-sampling approach followed in this work is the “uniform sampling” filtered developed by R. B. Rusu and explained in [79]. The algorithm creates a 3-D voxel grid over the whole point cloud and approximates each voxel with its centroid. Using the voxels' centroid is slower but more accurate than using their centre [79]. This method is also known as the “Voxel Grid” method and is known to be sensitive to noisy data, that can cause the output point cloud to not properly represent the original one [80]. Thus, the outlier removal algorithm will be of great importance and should be carried out before point cloud down-sampling.

3.4. Conclusions

This Chapter contextualized the data pre-processing operations carried out in this work. For this purpose, some basic notions and definitions of camera systems and image formation were first presented. Secondly, the image processing pipeline defined within the I3DS project was defined, as

well as the main sources of image distortion and how to correct them. Monocular images pre-processing operations include vignetting correction, lens distortion correction, bilateral filtering and histogram equalisation. Stereo pre-processing involves the additional stereo pair rectification and matching operations that allow to retrieve depth information from the images.

Finally, pre-processing of point clouds was addressed, with particular focus on LiDAR data. Point cloud down-sampling to make the data more easily handled by the posterior navigation algorithms and outlier removal to minimise the detrimental effect that spurious features can have on the accuracy of the navigation solution are the solution presented. Typical sources of distortion for point clouds were also described.

Chapter 4 – Navigation algorithms

Once the data acquired by the sensors have been pre-processed as described in Chapter 3, they will be input to the navigation solution. The aim of these algorithms is to determine the relative pose (i.e. relative position and orientation) of the target satellite with respect to the chaser satellite on-board which the INSES developed within the I3DS project will be mounted.

Both 2-D images from the stereo and the high-resolution cameras as well 3-D point clouds, either acquired by the LiDAR sensor or derived from the 2-D images, are considered within this dissertation. As such, this Chapter will present the algorithms necessary to retrieve relative navigation information from these different data. To this aim, Section 4.1 will introduce the Stereo-camera-based navigation method used in this research. Section 4.2 will describe then the navigation solution for the high-resolution camera. Finally, Section 4.3 will define the different algorithms that are considered for point cloud-based navigation.

4.1. Stereo navigation

The aim of visual tracking is to derive the relative motion (i.e. change in position and orientation) of an object with respect to the available images. This is done based on some observable property of the target object, with physical appearance the most commonly used in the literature. [81] categorises tracking algorithms as those that assume the object to be unchanged or that the change can be explained by some motion models, those that incorporate the dynamic model into the motion determination problem and those that try to include even more information about the motion process in the problem, such as adaptive speed or trajectory prediction models. The widely used KLT algorithm, also implemented in this dissertation and described in Chapter 3, is an example of algorithms in the first category [81].

The Stereo-camera-based navigation solution implemented in this work is represented in Figure 4.1.

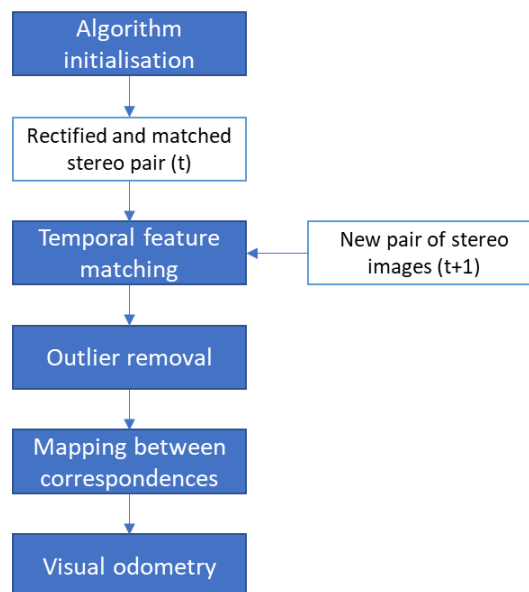


Figure 4.1: Stereo-camera-based navigation pipeline.

The input to the stereo navigation algorithm is a pair of undistorted and rectified stereo images previously matched using a KLT feature detector and tracker (see Chapter 3, Sections 3.2.2.1 to 3.2.3.2). Calling t the discrete time of the initialization step, once a new stereo pair is available, at time $t + 1$, features in the left-camera image at time t are tracked with the KLT tracker in the left-camera image at time $t + 1$. Simultaneously, features in the right-camera image at time t are tracked with the KLT tracker in the right-camera image at time $t + 1$.

This operation returns two sets of features, one in the left-camera image and one in the right-camera image, at time $t + 1$. Stereo matching between these sets of features is obtained by verifying the stereo constraint in that matching features are on the same epipolar line. Matches that do not comply with this are discarded in an outlier rejection operation. Since the features are tracked from time t to time $t + 1$, the number of correspondences M at the two time-steps is:

$$M_{t+1} \leq M_t \quad (4.1)$$

Therefore, the stereo-matched pairs at time t are further refined, by discarding the ones that have no match at time $t + 1$.

Once the correspondences are obtained, and given the camera calibration parameters, it is possible to obtain a non-linear mapping of the point correspondence h between the image at time t and the image at time $t + 1$ via the trifocal constraint. If the time between the two pair of images is known, then, the motion is retrieved by applying an Extended Kalman Filter strategy [82]. Under the assumptions that the noise is zero-mean white noise and that the system noise and measurement noise are uncorrelated, the instantaneous ego-motion state (i.e. 3-D motion of the camera) expressed in:

$$e_m = (V_X, V_Y, V_Z, \omega_X, \omega_Y, \omega_Z) \quad (4.2)$$

where V is the velocity and ω is the angular velocity, are integrated and filtered with the measurements h , to retrieve the pose, i.e. rotation R and translation T .

The entire procedure to obtain stereo visual odometry (i.e. temporal change in position and orientation) is also summarized in Figure 4.2. The final output of the stereo navigation is the relative translation along the three principal axes and the relative orientation, as a unit quaternion, of the target with respect to the first left-camera frame. Therefore, if distance and orientation between camera, and hence chaser, and target is known at an initial time, it is possible to obtain a relative pose at each posterior time.

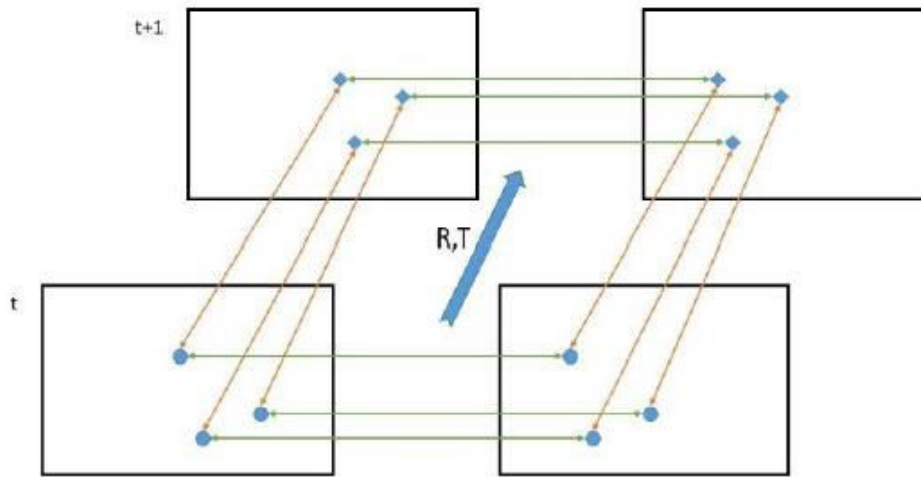


Figure 4.2: Scheme for stereo pose estimation. The matched features at time t are tracked at time $t+1$ and Pose Estimation in terms of rotation and translation is obtained.

4.2. Monocular navigation with fiducial markers

Scenarios of rendezvous with both cooperative and non-cooperative targets are contemplated within the I3DS project. In particular, the case of monocular navigation assumes that the target spacecraft is a cooperative target. When dealing with cooperative targets, an efficient strategy involves the detection of fiducial markers, whose three-dimensional position in the target reference frame is known. Fiducial markers are patterns that are robustly detectable with a clearly defined centre point, e.g. a 2x2 checkerboard, and placed at known locations. An example of fiducial markers on the simulated images used in this work and that will be introduced in the following chapter is shown in Figure 4.3.

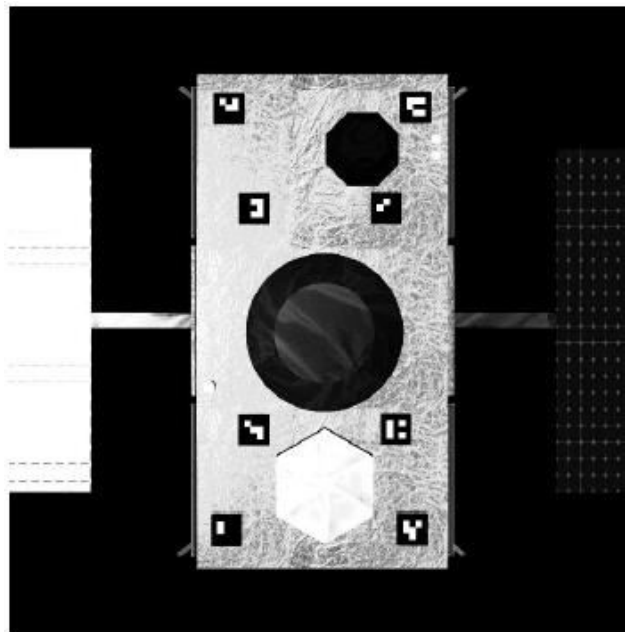


Figure 4.3: ArUco custom fiducial markers on an image of the target spacecraft simulated with SPICAM software by Thales Alenia Space France (TAS-F).

The proposed algorithm makes use of two custom dictionaries of fiducial markers, based on ArUco libraries [83]. One dictionary is for medium-range operations while the other is for close-range operations. The core of the navigation solution is the Perspective-n-Point (PnP) algorithm [68]. The general formulation of the PnP problem requires finding the transformation T_k that minimizes the image re-projection error:

$$\min_{T_k} \sum_{i=1}^N \|p_k^i - \hat{p}_{k-1}^i\|^2 \quad (4.3)$$

where p_k^i is the position of feature i in the image I_k , $\hat{p}_{k-1}^i$ is the reprojection of the 3-D point X_{k-1}^i into image I_k according to the transformation T_k . The minimal case involves four 3-D-to-2-D correspondences to provide an unambiguous solution.

The algorithm is initialized when an image is received from the monocular camera. A scale-invariant and rotation-invariant template matching finds the fiducial markers from the first dictionary in the image and establishes a set of Regions of Interest (ROIs) around them [84] [55]. Once the ROIs are set, the centre of each marker is detected at sub-pixel accuracy. If at least four markers have been detected, their 2-D position in the image is matched with the corresponding 3-D position and the PnP algorithm provides the pose estimate.

Then the navigation filter searches for fiducial markers from the second dictionary in the same fashion. It is worth noting that if no markers from the first dictionary are found, then relative pose is computed with the second set of fiducial markers. If, on the contrary, pose is already detected with first set of markers and second-dictionary markers are detected, the already available pose is refined. The entire HR navigation pipeline is depicted in Figure 4.4.

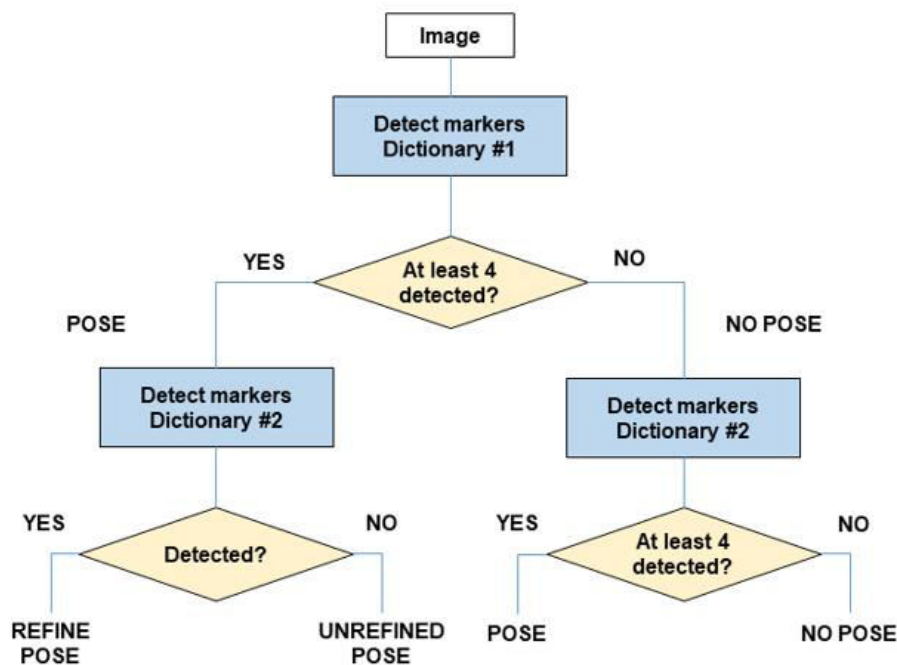


Figure 4.4: HR-camera-based navigation pipeline.

Since there is direct correspondence between 2-D and 3-D points and the camera parameters are known, the output of the monocular navigation with fiducial markers is the relative position and orientation of the target reference frame with respect to the camera reference frame.

4.3. Point cloud navigation

In this work, the only point cloud navigation that has been implemented is LiDAR navigation. However, image-based point clouds could also be used as input to these navigation algorithms, on their own or combined with LiDAR-based point clouds.

GPS and IMUs are frequently used to calculate displacements. However, they have the disadvantages of not being accurate enough to deal with precise positioning and not being available or precise (GPS) in some scenarios such as the interior of buildings. Therefore, point cloud registration has been extensively studied, with many methods present in the literature, as a way of dealing with the positioning and relative-pose estimation problems [85].

Since its first appearance in a 1992 paper by Besl and McKay [86], the Iterative Closest Point (ICP) algorithm was established as the most widely used point cloud registration method. In its initial form, ICP was proposed as a method for point cloud registration but also for free-form curve matching. This work was relatively novel as a 3-D free-form registration example since there had not been significant research in that direction [86]. Its main advantages are easy computation, fast implementation and simplicity, together with fast and monotonic convergence [86]. Its main drawbacks are the assumption of total overlap between the source or data point cloud and the target or model point cloud, the theoretical requirement that the point clouds are taken from a known geometric surface instead of measured [85] [86] and its instability with noisy data [87]. Total overlap refers to every point in the source point cloud having a match in the target point cloud. Furthermore, although the algorithm converges monotonically to a local minimum, it may not converge on the global minimum as desired [86].

Many methods and modifications to the ICP algorithm have been proposed to try to deal with these limitations. These ICP variants are different from one another in the way they perform each step of the general algorithm, i.e. selection of the set of points to match from the two point clouds, matching between them, weighting of the corresponding points, rejection of certain pairs, definition of a registration error metric and minimization of the error metric [88]. Some of these variants include the “point-to-plane” ICP [89], the Expectation Maximisation ICP (EM-ICP) [90], Color ICP [91] and non-linear ICP [92], among many others.

The methods evaluated in this dissertation are the classical Iterative Closest Point (ICP) [86], the ICP modifications Generalized ICP (GICP) [85] and Trimmed ICP (TrICP) [87] and the Histogram of Distances (HoD) [93] point descriptor with Kalman Filter (KF)-based point cloud registration [94].

4.3.1. Iterative Closest Point

The general flow of the ICP algorithm is shown in Figure 4.5. The algorithm iteratively searches for the best transformation between two point clouds until a stopping criterion is met.

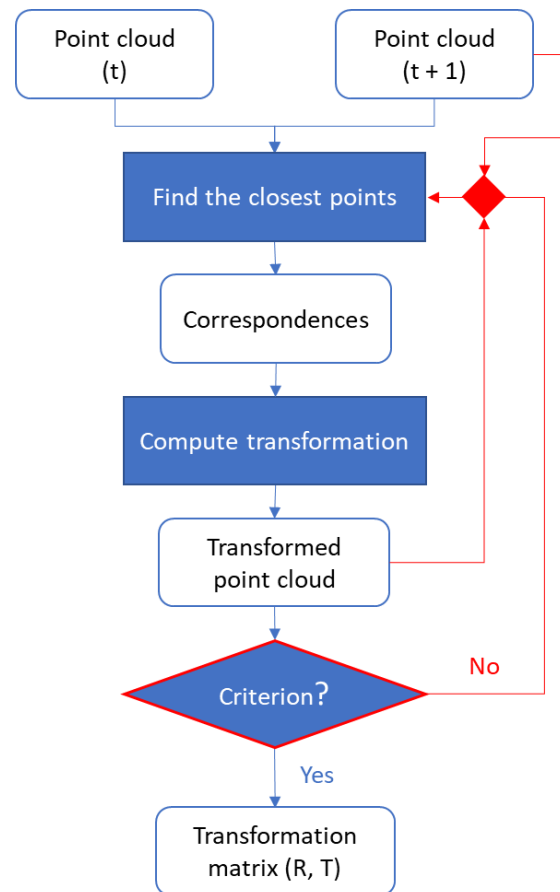


Figure 4.5: Block diagram of the general ICP navigation algorithm.

The first step of the algorithm involves establishing correspondences between points in the two point clouds. The correspondence is obtained by finding the nearest neighbour of a point in the first point cloud within the second point cloud, and it is calculated via a k -d tree.

The concept of k -d trees (from k -dimensional trees) or multidimensional binary search trees was introduced in 1975 by Bentley [95] as a “data structure for storage of information to be retrieved by associative searches.” In this structure, every node represents a k -dimensional data point and every non-leaf node, a hyperplane separating the points in two categories every time. Because of its efficiency in storage requirements and its fast implementations, this method has been widely used in the literature. A thorough definition of the method can be found in [95].

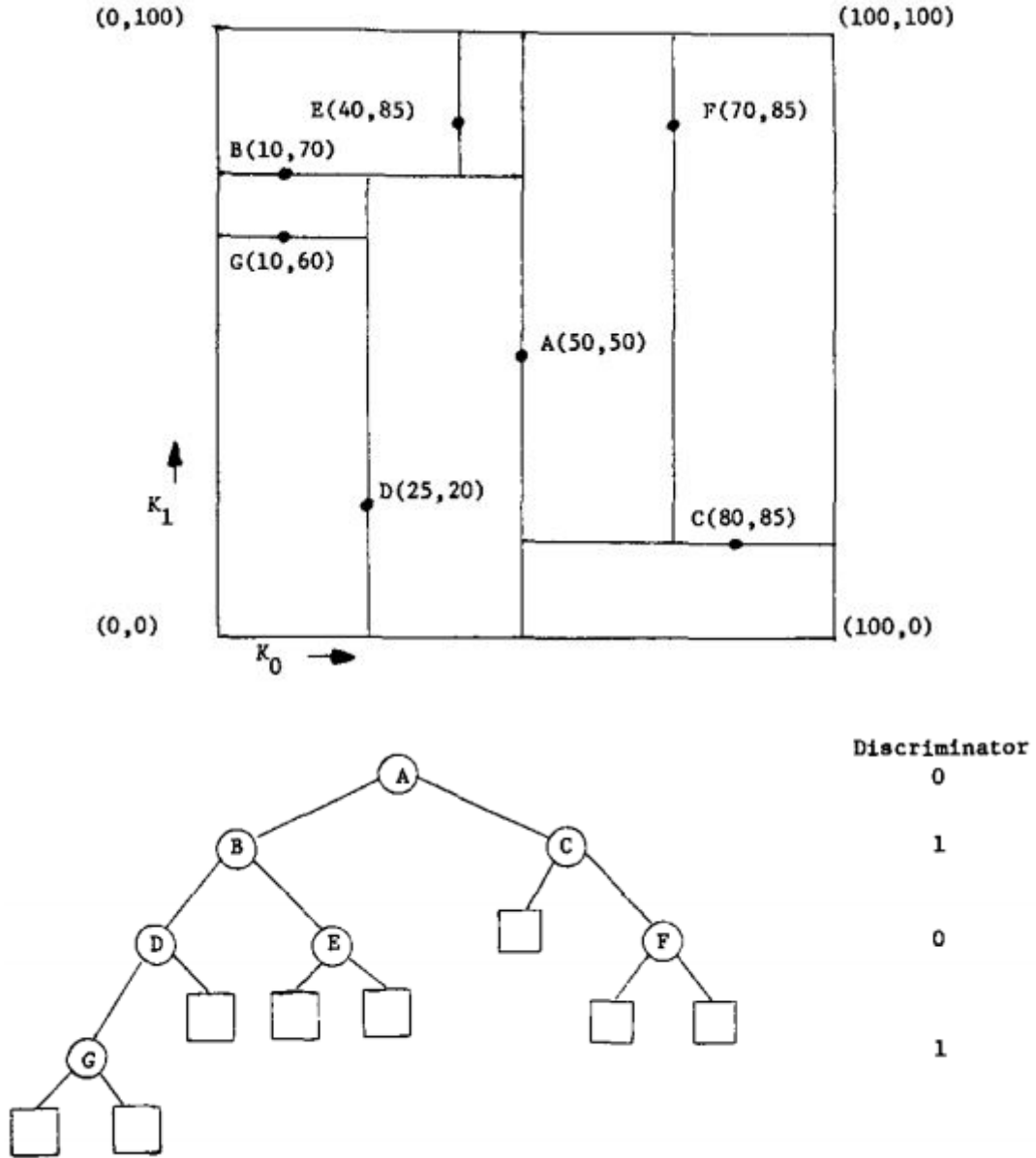


Figure 4.6: Example of a 2-d tree or k-d tree of dimension 2 [95].

The problem can then be stated as follows: given two point clouds, $P = \{p_1, \dots, p_n\}$ the target point cloud, and $Q = \{q_1, \dots, q_n\}$ the source point cloud, a transformation (with a translation component T and a rotation component R) is searched to minimise the mapping error:

$$E(R, T) = \sum_{i=1}^{N_p} \sum_{j=1}^{N_q} w_{i,j} \|p_i - (R q_j + T)\|^2 \quad (4.4)$$

where N_p and N_q are the number of points in the two point clouds. The values $w_{i,j}$ represent the weights of the correspondence between the point p_i in the target point cloud and the point q_j in the source point cloud. If the correspondence is verified, then $w_{i,j} = 1$, otherwise $w_{i,j} = 0$.

The search for the best transformation matrix between the corresponding points in the two point clouds is done by means of Singular Value Decomposition (SVD) as explained in [96]. In this aspect the algorithm used here differs from the original ICP proposal [86] in which the optimal transformation was computed using a quaternion-based algorithm.

Finally, the transformation matrix is applied to the source point cloud. The algorithm iteratively calculates the transformation matrix between the last estimated point cloud and the target point cloud until one of the stopping criteria is met. [86] considered only a termination criterion based on the change in mean square error between successive registrations. However, three different stopping criteria are considered in the implementation done in this work [97]:

- a) A defined maximum number of iterations is performed.
- b) The algorithm converges: The difference between the previous transformation and the current estimated transformation is below a certain threshold.
- c) The algorithm finds a solution: The sum of square errors between the transformed point cloud and the target point cloud is below a certain threshold.

However, because ICP can converge to local minima and thus find incorrect correspondences, which can negatively affect the registration accuracy, some action must be taken to discard these wrong correspondences [97]. In this work, a maximum distance between points to consider them to be a good correspondence is defined.

4.3.2. Generalized Iterative Closest Point

Halfway between the classical ICP and the fully probabilistic point cloud registration models that incorporate some relationship between the source and target point clouds, the Generalised Iterative Closest Point (GICP) algorithm appeared. GICP is motivated by both the classical ICP and the “point-to-plane” ICP derivation that includes structural assumptions in the registration problem by not penalizing offsets along a surface [85].

GICP is based on using Maximum Likelihood Estimation (MLE) as the non-linear optimization step and computing point correspondences between the two point clouds through k -d trees. The method introduces symmetry in that it uses structural information from both point clouds, which improves accuracy, and assumes the scene objects to be locally planar. Furthermore, by including structural information from both point clouds the influence of incorrect matches is diminished and so, the election of the algorithm’s parameters is not as critical as with other versions of ICP [85].

The motivation to include the GICP algorithm into this research was based on the realisation that the morphology of a spacecraft is generally composed of planes, such as the solar panels or the different faces of the satellite. An example of this is shown in Figure 4.7. Thus, the assumption of locally planar surfaces made by GICP is relevant and incorporating this shape information into the registration method should increase the registration performance. Furthermore, since the two point clouds are representations of an object measured by the LiDAR sensor in two different moments in time and from two different perspectives, in practice the point correspondences will not be exact (i.e. it is very unlikely that the exact same point is sampled in the two different point clouds). GICP takes this into consideration by deriving only a constraint along the surface normal of each point, instead of searching for exact correspondences [85].

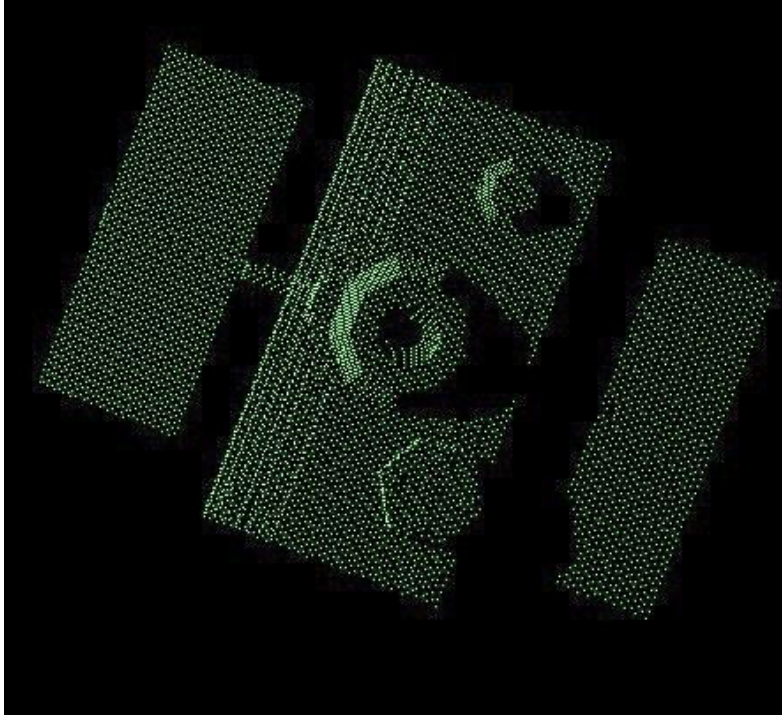


Figure 4.7: Example of the point cloud data simulated by Thales Alenia Space - France (TAS-F) used in this work. It can be observed how most of the spacecraft morphology is formed by planes, which motivated the use of the GICP point cloud registration algorithm.

[85] expresses the minimisation problem to compute the transformation matrix T in the classical ICP algorithm as:

$$T \leftarrow \arg \min_T \{ \sum_i \omega_i \|T \cdot b_i - m_i\|^2 \} \quad (4.5)$$

where T is the transformation matrix, ω_i the weight associated to each correspondence, b_i a point from the target point cloud and m_i the closest point to b_i after applying the transformation to the source point cloud at each iteration step.

On another hand, the point-to-plane variant of ICP incorporates the surface normal information into the minimisation problem as seen below:

$$T \leftarrow \arg \min_T \{ \sum_i \omega_i \|\eta_i(T \cdot b_i - m_i)\|^2 \} \quad (4.6)$$

Thus, instead of minimising $\sum \|T \cdot b_i - m_i\|^2$, the point-to-plane algorithm tries to minimise the registration error along the surface normal η_i by minimising the projection of $(T \cdot b_i - m_i)$ onto the sub-space spanned by η_i .

Finally, GICP incorporates a probabilistic model into this minimisation step (Equations (4.5) and (4.6)). To do so, the algorithm assumes that the source and target point clouds $A = \{a_i\}$ and $B = \{b_i\}$ are derived from an underlying set of true points (perfect correspondences) $\hat{A} = \{\hat{a}_i\}$ and $\hat{B} = \{\hat{b}_i\}$ following the Gaussian distributions:

$$a_i \sim \mathcal{N}(\hat{a}_i, C_i^A) \quad (4.7)$$

$$b_i \sim \mathcal{N}(\hat{b}_i, C_i^B) \quad (4.8)$$

where $\{C_i^A\}$ and $\{C_i^B\}$ are the covariance matrices associated with the measured points. Following this reasoning, the new minimisation problem results [94]:

$$T = \arg \min_T \sum_i d_i^{(T)^T} (C_i^B + T C_i^A T^T)^{-1} d_i^{(T)} \quad (4.9)$$

with:

$$d_i^{(T)} = b_i - T a_i \quad (4.10)$$

Expressing the problem in this way gives more freedom to model the situation, as the covariance matrices can be chosen. For example, the classical ICP algorithm can be reached by modelling the covariances as:

$$C_i^B = I \quad (4.11)$$

$$C_i^A = 0 \quad (4.12)$$

and the point-to-plane algorithm results:

$$C_i^B = P_i^{-1} \quad (4.13)$$

$$C_i^A = 0 \quad (4.14)$$

As previously stated, GICP assigns to each point a constraint along its surface normal, that is modelled in the covariance matrices. In particular, each sampled point is considered to be distributed with high covariance along its local plane and very low covariance along its normal direction, which corresponds to knowing its position along the normal with very high confidence but not its location in the plane. If the covariance along the normal of a point is represented by a small constant ϵ , this results in the following covariance matrix [85]:

$$\begin{pmatrix} \epsilon & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (4.15)$$

Thus, the covariance matrices C_i^B and C_i^A can be computed by rotating the covariance matrix in Equation (4.15) so ϵ represents the covariance along the surface normals on the points b_i and a_i , represented by μ_i and ν_i :

$$C_i^B = R_{\mu_i} \cdot \begin{pmatrix} \epsilon & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot R_{\mu_i}^T \quad (4.16)$$

$$C_i^A = R_{\nu_i} \cdot \begin{pmatrix} \epsilon & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot R_{\nu_i}^T \quad (4.17)$$

4.3.3. Trimmed Iterative Closest Point

One of the main drawbacks of the classical ICP algorithm is the assumption of total overlap between point clouds being registered [85]. Based on this hypothesis, the algorithm search for correspondences for all points in the source point cloud. However, in practice, situations such as

occlusions can lead to mismatches when parts of the object that the source point cloud represent are not found in the target point cloud. In space, this can occur in the final approximation phase of a rendezvous, when the distance between the chaser and target spacecrafts is so small that a part of the target spacecraft “leaves” the chaser’s FOV. Additionally, thin elements in the target spacecraft such as antennae, can have an effect of appearing and disappearing between laser scans as while the chaser spacecraft is moving close to the target satellite and facing it, it can occur that in some scans there is no laser beam that reflects on them. Figure 4.8 depicts both effects. For simplicity, the target and chaser spacecrafts are represented as boxes and in Figure 4.8.b, the vertical density of laser beams (vertical angular resolution of the LiDAR sensor) is lower than in practice.

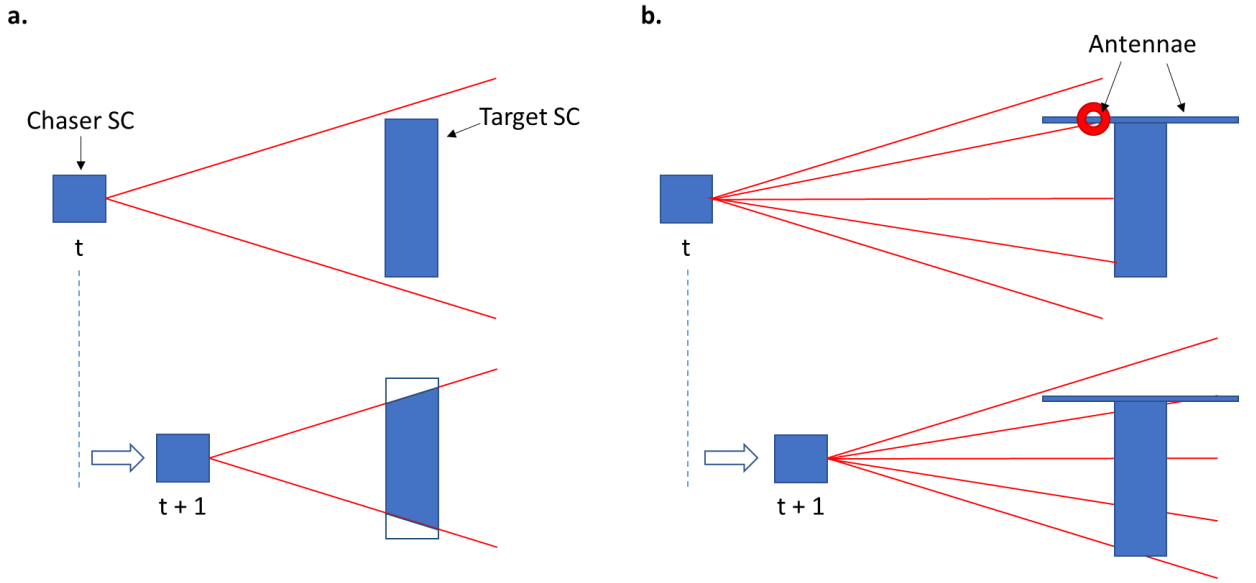


Figure 4.8: Possible point cloud changes in space. (a) Part of the target spacecraft disappears from the field of view of the chaser and (b) thin elements such as antennae can appear and disappear between laser scans.

Therefore, a potential solution to the non-total overlap problem was included in this work. This is the Trimmed ICP (TrICP) algorithm [87], a modification of the classical ICP that takes into consideration the non-full correspondence between point clouds.

TrICP is a fast point cloud registration method, robust to noise and applicable to overlaps under 50 % [87]. It is based on the steady use of the Least Trimmed Squares (LTS) method [98] in all major operation phases: to cope with noise or partial overlap, to estimate the optimal transformation at each iteration and to produce the global cost function to be minimised. It Was developed from RICP, an ICP variation that deals with incomplete point cloud data for point cloud registration [87].

TrICP considers the source and target point clouds, denoted as $P = \{p_i\}_1^{N_p}$ and $M = \{m_i\}_1^{N_m}$, with N_p representing the number of points in P and N_m the number of points in M . Its main contribution is to assume that $N_p \neq N_m$ and that there is a minimum certain proportion of points from the source point cloud (N_{p0}) that can be mapped into the target point cloud. This proportion is the *minimum overlap* ε and relates the two point clouds as [87]:

$$N_{p0} = \varepsilon N_p \quad (4.18)$$

The method requires however that the overlapping part of the two point clouds is characteristic enough (i.e. not symmetric or featureless) to allow for unambiguous correspondence.

The different steps of TrICP are:

1. For all points in P , find the closest point in M and compute the squared of their individual distances d_i :

$$d_i = \|m_{cl} - p_i\|^2 \quad (4.19)$$

with:

$$m_{cl} = \arg \min_{m \in M} \|m - p_i\| \quad (4.20)$$

2. Organise d_i^2 in descending order, select the N_{p0} least values and compute their sum S'_{LTS} .
3. If any stopping criteria is met, exit the algorithm. Otherwise do $S_{LTS} = S'_{LTS}$. The stopping criteria are the standard criteria defined in [99], based on the number of iterations and the minimum square error (MSE) for the subset N_{p0} , e :
 - a. A certain maximum number of iterations is met.
 - b. The trimmed MSE e is below a certain threshold.
 - c. The change in $e = \frac{|e-e'|}{e}$ in successive iterations is below a certain threshold.
4. For the pairs selected in N_{p0} , compute the optimal transformation. This will be the transformation that minimises the sum of least trimmed squares S'_{LTS} .
5. Apply the computed transformation to all points in P as:

$$p_i(R, t) = R p_i + t \quad ; \quad P(R, t) = \{p_i(R, t)\}_1^{N_p} \quad (4.21)$$

and re-iterate from step 1.

As classical ICP, TrICP always converged monotonically to a local minimum, however convergence to the global minimum depends on the initialisation [87].

4.3.4. Histogram of Distances point descriptor with Kalman Filter-based registration

The last considered method involves assigning HoD descriptors [93] to all the points in the point cloud at discrete time t and to all the point in the point cloud at time $t+1$. Then, correspondences between the descriptors are sought and thus point-cloud registration is performed. The point cloud registration is finally achieved by means of a Kalman Filter [94].

4.3.4.1. Histogram of Distances (HoD) descriptor

Developed at Cranfield University, the Histogram of Distances (HoD) point descriptor was created to minimize the computing load and therefore the computing time associated with most local descriptors. This is done by eliminating the need to compute a reference frame or axis (LRF/A) on which the descriptors are then usually defined. Besides reduction of computing time, HoD also achieves higher robustness to noise and non-uniform subsampling when compared with other standard methods [93].

Given a point cloud $P = \{p_i\}^K$, with K being the total number of points in P , the HoD descriptor is computed for each keypoint in the following steps:

1. A spherical volume delimited by a sphere of radius r centred in the keypoint is created. Keypoints are assumed to have been previously extracted.
2. A reference point p_r situated on the border of the spherical volume (i.e. on the sphere) is selected.
3. For all the points confined in the spherical volume, the L2-norm between p_r and each point is computed as:

$$d_i = \|p_r - p'_i\|^2 \quad (4.22)$$

4. To make the descriptor invariant to noise, the d_i distances are normalised to the round-floor values d'_i , that will be encoded in the HoD, as:

$$d'_i = \left\lfloor \frac{d_i}{\max(d_i)} B \right\rfloor \quad (4.23)$$

where B is the bin size of the HoD.

To increase the robustness to the point cloud resolution, the HoD descriptor is normalised to sum up to 1. Furthermore, to achieve mesh resolution invariance the value of the support radius r is chosen as the multiple of each scene's mesh resolution mr . Finally, the descriptiveness of HoD is increased by defining two histograms for each keypoints, each of them with a different bin size B (one fine and one coarse).

These steps are represented in Figure 4.9.

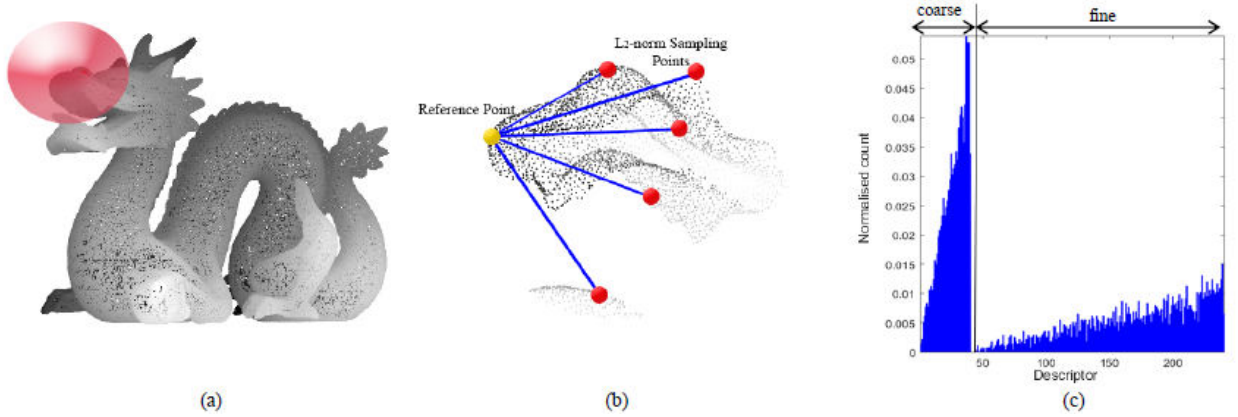


Figure 4.9: Histogram of Distances (HoD) concept. (a) A spherical area of radius r (red) centred on a keypoint is extracted. (b) One random border point from the local area is selected as reference point (yellow) and the reference point to vertices L2-norms distances are calculated (in red as example). (c) Coarse and fine normalised distances encoded into a Histogram of Distances [93].

4.3.4.2. Point cloud matching

Once the point descriptors have been assigned, correspondences between the source and target point clouds are established using the Fast Library for Approximate Nearest Neighbours (FLANN), described in Chapter 3.

4.3.4.3. General Kalman filter

The Kalman Filter (KF) is a method to define optimal state estimators used when the variables of interest can only be measured indirectly or when measurements are available from different sensors but may be subject to noise [100].

Figure 4.10 shows the general flow of Kalman filter and the concept of a state estimator.

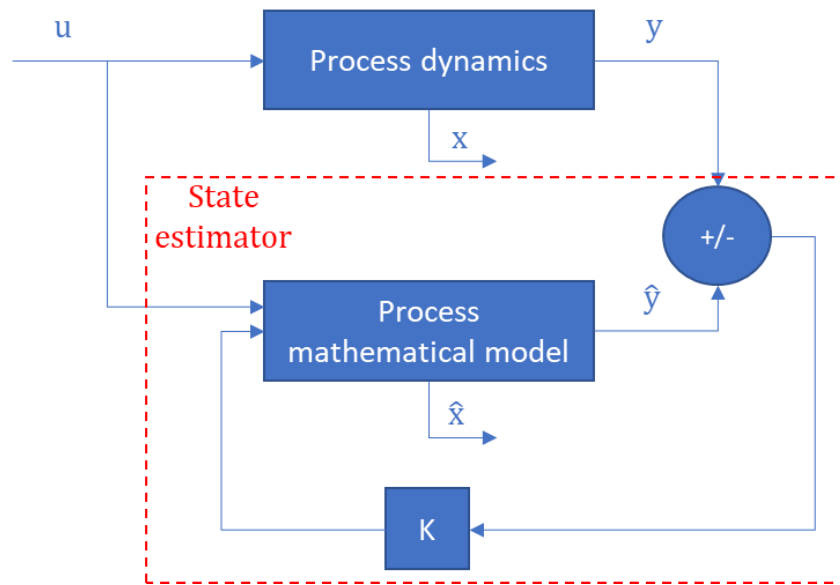


Figure 4.10: General flow of Kalman filter and concept of a state estimator.

In the figure, u represents the input that through a process of unknown dynamics produces an output measurement y and a state variable that cannot be directly measured x but whose value is searched. However, the process can be modelled mathematically to derive an estimate of the measurement $\hat{y}$ and an estimation $\hat{x}$ of the system state x . Comparing then the measured variable y with the output of the mathematical model $\hat{y}$, a refinement to the mathematical model can be introduced through a control variable K with the objective of minimising the difference between the process dynamics and the process mathematical model. The optimal solution would then correspond with the best estimation of the searched state x .

In mathematical terms and with the incorporation of noise, the Kalman filter can be represented as:

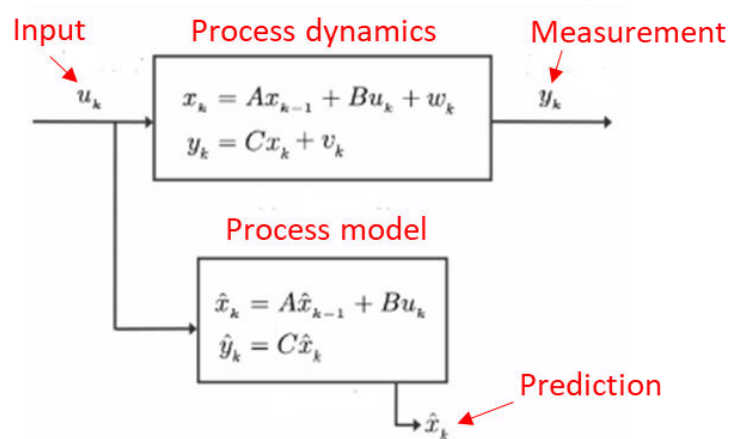


Figure 4.11: KF diagram in mathematical terms and with the incorporation of noise [100].

In Figure 4.11, ω_K represents the process noise and v_K , the measurement noise. Both variables are random and can be approximated by a Gaussian distribution as:

$$\omega \sim \mathcal{N}(0, Q) \quad (4.24)$$

$$v \sim \mathcal{N}(0, R) \quad (4.25)$$

Kalman filter combines the prediction $\hat{x}_k$ and the measurement y to produce an optimal estimate in the presence of measurement and process noise [100]. In terms of probability density functions, it can be schematised as shown in Figure 4.12. Having an initial state estimate $\hat{x}_{k-1}$, distributed with a certain variance or uncertainty, the variance P_k^- of a newly predicted state estimate $\hat{x}_k^-$ will be larger due to process noise. Additionally, the measurement y_k at the same time stamp will also conform to a Gaussian distribution with a variance due to the measurement noise. Finally, the optimal state estimate $\hat{x}_k$, at that time instant will result from the multiplication of the predicted state estimate and the measurement distributions. The resulting variance P_k will be smaller than the other distributions' variances [100].

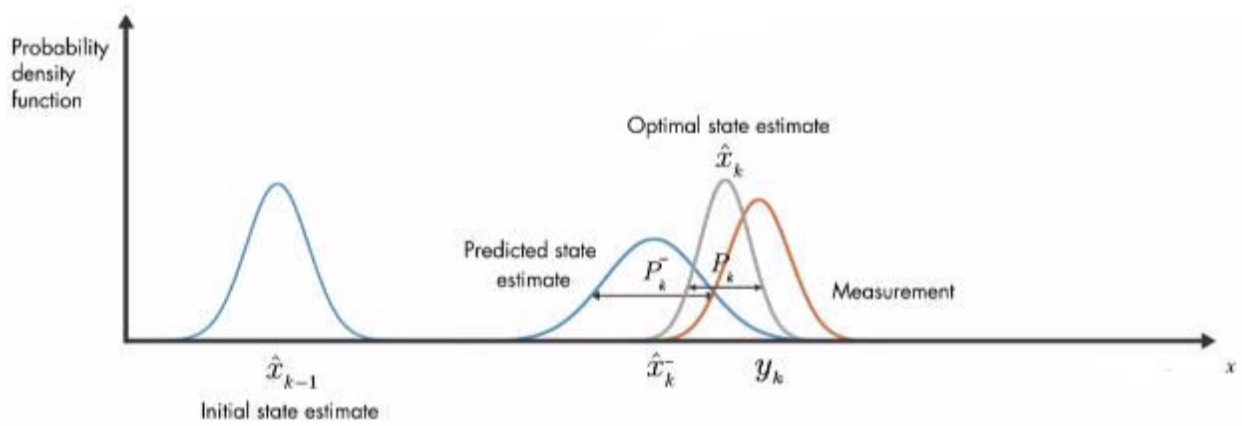


Figure 4.12: Kalman filter state estimator explained in probability density functions [100].

The Kalman filter equation can be expressed as:

$$\hat{x}_k = \hat{x}_k^- + K_k(y_k - C \hat{x}_k^-) \quad (4.26)$$

with:

$$\hat{x}_k^- = A \hat{x}_{k-1} + B u_k \quad (4.27)$$

where $\hat{x}_k$ is the “a posteriori estimate” (i.e. the grey distribution in Figure 4.12) and $\hat{x}_k^-$ is the “a priori estimate”, done before the measurement is taken (i.e. the blue distribution with the largest variance in Figure 4.12) and derived from the previous estimate $\hat{x}_{k-1}$ and the current input u_k . The second part of the right term of the equation, $K_k(y_k - C \hat{x}_k^-)$, uses the measurement y_k to incorporate it into the prediction to update the a priori estimate. Therefore, the KF can be divided into two major stages: prediction and update shown in Figure 4.13.

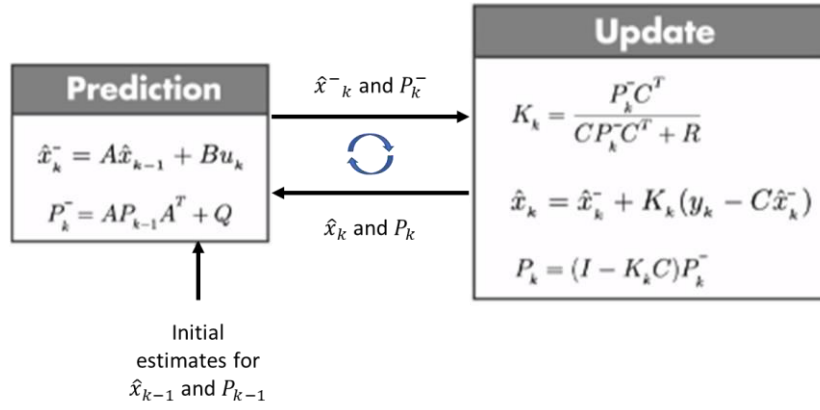


Figure 4.13: KF is divided into two major steps: prediction and update. Information sourced from [100].

In the update state, the Kalman gain K_k is introduced with aim to minimise the posterior error covariance P_k . It determines how heavily the measurement y_k and the a priori estimate $\hat{x}_k^-$ contribute to the a posteriori estimate $\hat{x}_k$. In this way, if the measurement noise is small, the measurement is stressed more and contributes more to the computation of $\hat{x}_k$. If on the contrary the error of $\hat{x}_k^-$ is small, it is this term which contributes more [100].

Overall, the main advantage of KF is that it does not need all the past information to compute the current state. Instead, the algorithm only needs the previous estimate $\hat{x}_{k-1}$ with its covariance P_{k-1} and the current sensor measurement y_k [100].

4.3.4.4. Application of the Kalman filter to point cloud registration

Once the correspondences between the source and target point clouds have been established, point cloud registration is performed using a Kalman filter. In particular, a modification of the KF-based registration presented in [94] is applied here.

In this implementation, the variables sought for are the rotation matrix R and the translation vector T to minimize the registration error (Equation (4.4)). In order to include the rotation matrix and translation vector into the KF process, the state vector is defined as the concatenation vector of the rotation values and the translation values as:

$$\hat{s}^T = \begin{bmatrix} (R_{11}, R_{12}, R_{13})^T \\ (R_{21}, R_{22}, R_{23})^T \\ (R_{31}, R_{32}, R_{33})^T \\ (T_X, T_Y, T_Z)^T \\ (\dot{T}_X, \dot{T}_Y, \dot{T}_Z)^T \\ (\ddot{T}_X, \ddot{T}_Y, \ddot{T}_Z)^T \end{bmatrix} \quad (4.28)$$

In addition, linear velocity and acceleration are added to increase accuracy.

The prediction state equation has no input control and the observation matrix is in the form of:

$$H = \begin{bmatrix} m & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & \Delta t^2 & 0 & 0 \\ 0 & m & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & \Delta t^2 & 0 \\ 0 & 0 & m & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & \Delta t^2 \end{bmatrix} \quad (4.29)$$

where m is the measurement and Δt is the time between two measurements. The KF process can be started to obtain the transformation matrix from the final state vector s .

The overall flow of the HoD+Kalman algorithm is represented in Figure 4.14.

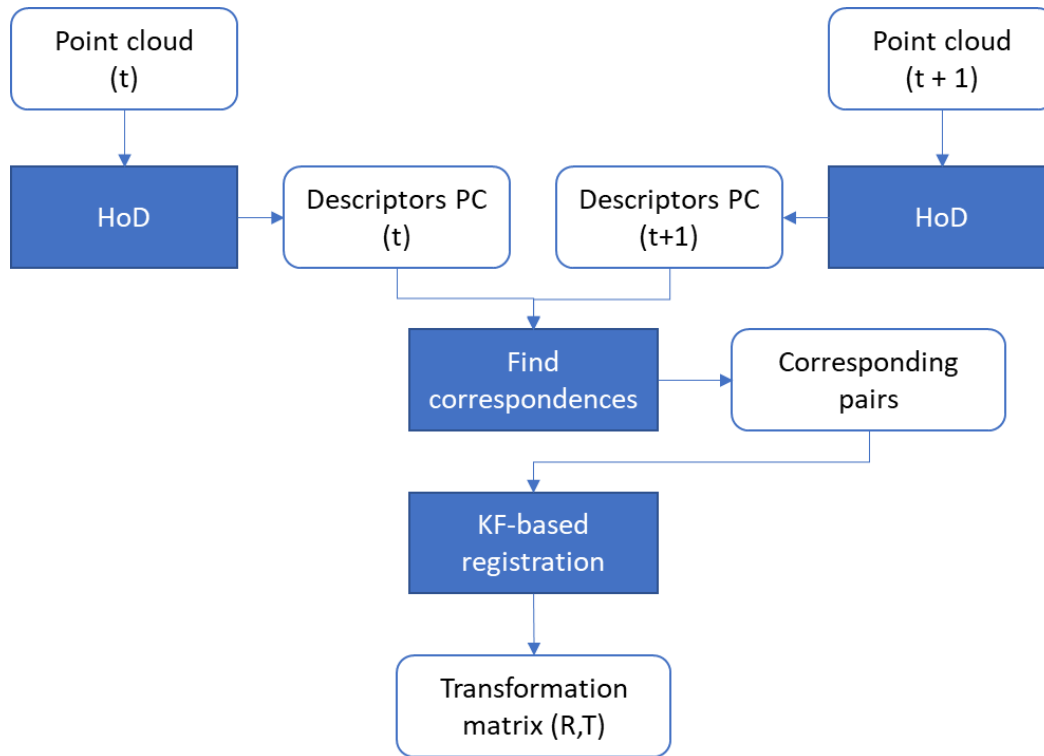


Figure 4.14: Block diagram of the HoD+Kalman navigation algorithm.

4.4. Conclusions

This Chapter has given a broad overview of the navigation methods traditionally used for the sensors analysed in this work, with special emphasis on the methods here employed, that have been further described.

Stereo-camera-based navigation has been done by means of a temporal matching of stereo pair from successive frames, previously rectified and matched for features. The widely used PnP algorithm has been applied to monocular-camera-based navigation, supported by two sets of known fiducial markers on the surface of the target spacecraft. Finally, different point cloud registration algorithms have been evaluated for LiDAR-based navigation, namely the classical ICP, Generalised ICP, Trimmed ICP and a Histogram of Distances point descriptor with a Kalman filter-based registration.

Chapter 5 – Tests and results

A thorough analysis on the speed performance of the proposed solutions has been carried out. The tests involved resizing the images and down-sampling the point clouds to assess how the time of processing is influenced. At the same time the aim was to understand if the overall accuracy could be degraded with less resolution in the raw and pre-processed data.

The test bench to simulate the system is formed of the Xilinx UltraZed-EG FPGA board and an 8x Intel® Core™ i7-6700 CPU @ 3.40 GHz desktop computer with Ubuntu 16.04.4 LTS to simulate the OBC. In particular, the FPGA board was used for the end-to-end image processing pipeline and the desktop computer for the end-to-end LiDAR data processing pipeline (i.e. pre-processing and navigation algorithms). This workflow is illustrated in Figure 5.1.

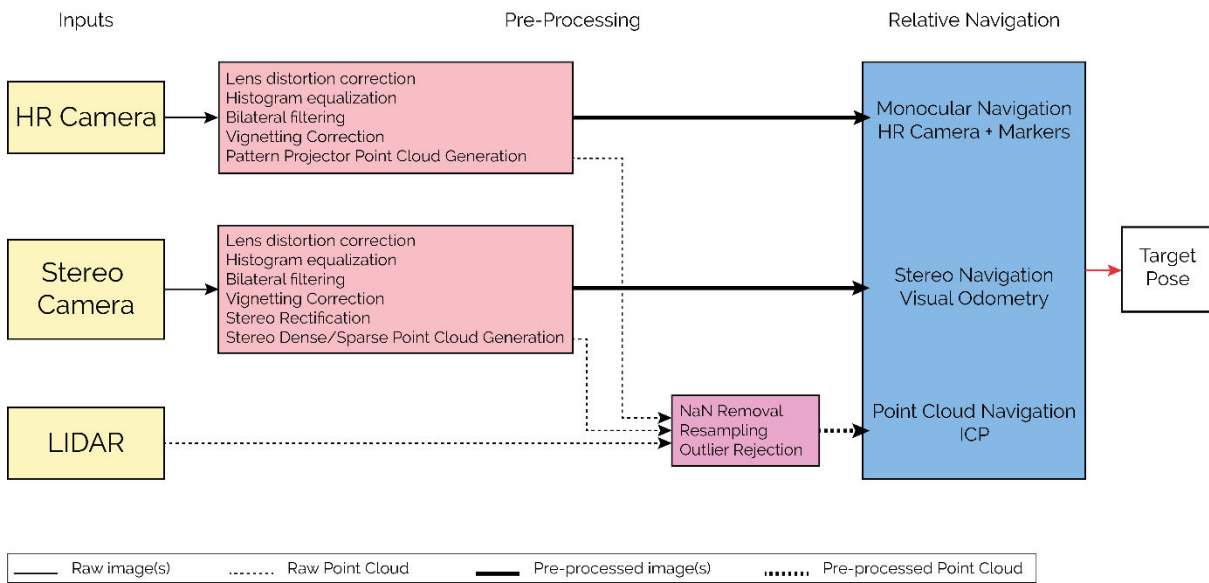


Figure 5.1: End-to-end processing pipeline, from data acquisition to navigation solution.

This Chapter will describe all the aspects relative to the performed tests. First, the utilised test data will be introduced. Secondly, the hardware solutions implemented to simulate the INSES hardware will be presented. Thirdly, a focus will be made on the software elements of the tests: The used libraries and software programs. Finally, the integration of all these components together will be defined.

5.1. Data

The tests have been carried out on simulated data for the three sensors. In particular, the high-resolution and stereo images have been simulated using the TAS-F internal SPICAM software. The LiDAR data was simulated by TAS-F using the free open-source software package BlenSor (Blender Sensor Simulation) [101]. The size of all raw images is 2048 pixels x 2048 pixels and the raw point clouds have 152100 points. Examples of the simulated data are shown in Figure 5.2.

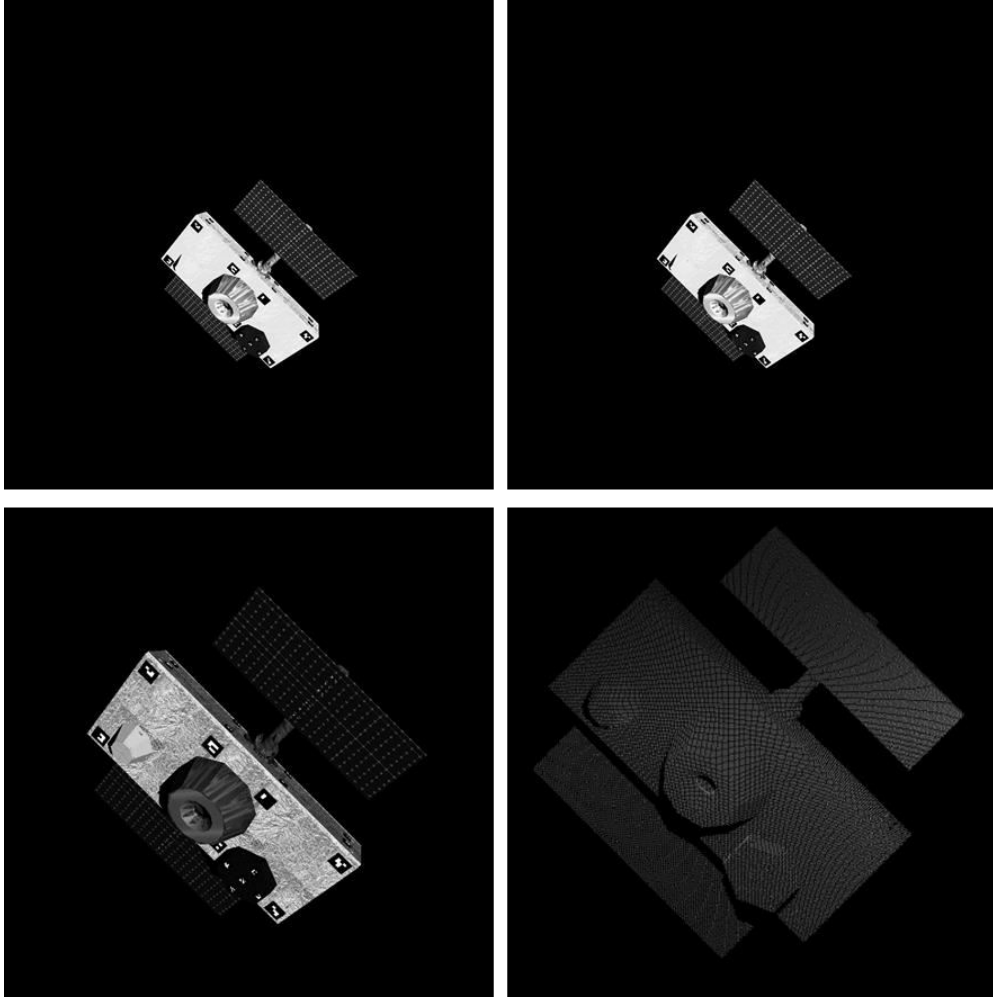


Figure 5.2: Examples of simulated data used in this work. From top to bottom and from left to right: Stereo data, HR data and LiDAR data. The images were simulated with SPICAM software and the LiDAR data, with BlenSor software.

The characteristics of the simulated high-resolution monocular camera and stereo-camera are listed in Table 5.1.

Table 5.1: Parameters of the simulated HR and stereo cameras.

	Stereo	HR
Focal length [mm]	12	21
Pixel pitch [μm]	5.5	5.5
Principal point x coordinate [pixels]	1024	1024
Principal point y coordinate [pixels]	1024	1024
Baseline [cm]	15	N.A.

The simulated trajectory is a straight-line approach trajectory representing the final phase of space rendezvous. From 20 m, the chaser spacecraft approaches the target spacecraft without any rotational motion. The approach velocity is 5 cm/s. The motion is along the camera z-axis and the

target z-axis is perfectly antiparallel aligned with the camera z-axis, therefore no yaw, pitch and roll are present.

The target satellite features fiducial markers on its surface to support visual navigation, especially in the case of monocular visual odometry as a means of overcoming scale ambiguity.

For the case of stereo-camera-based navigation, despite the solution being able to run at speeds greater than 10 Hz (see Chapter 6), a framerate of 10 Hz led to erroneous results. This is because at a speed of 5 cm/s and framerate of 10 Hz, the inter-frame motion would be 5mm, which is too small for the proposed stereo-camera to properly track the motion of features. Therefore, different framerates have been tested, namely 1Hz and 2Hz.

Furthermore, the testing included different image and point cloud dimensions (resolution) to assess the trade-off between computing time and navigation accuracy.

5.2. Hardware solutions for data processing

The OBC, on which the LiDAR processing pipeline will be implemented, has been simulated using an 8x Intel® Core™ i7-6700 CPU @ 3.40 GHz desktop computer with Ubuntu 16.04.4 LTS.

As for the FPGA that will be integrated in the I3DS INSES, it has been simulated by the UltraZed-EG FPGA manufactured by Xilinx [102]. The choice of this board was based on the fact that although this model is not space-rated, Xilinx also provide some space-rated FPGA boards for which the developing tools are the same as for the UltraZed-EG: the Virtex®-4QV and the Virtex®-5QV FPGA families [103].

The UltraZed-EG FPGA is a flexible System-On-Module (SOM) based on the Xilinx Zynq® UltraScale+™ Multi-Processor System-on-Chip (MPSoC). It is designed with a small form factor and it features Ethernet, USB, system memory and configuration memory needed for embedded processing. It is also provided with a Linux Board Support Package (BSP) [102]. The FPGA board contains Quad-core ARM Cortex-A53 MPCore at 1.2 GHz applications cores, Dual-core ARM Cortex-R5 MPCore at 500 MHz real-time cores and a Mali™-400 MP2 at 600 MHz graphics processor [104].

Some application examples of the UltraZed-EG include flight navigation, missile and munitions, military construction, secure solutions, networking, cloud computing security, data centres, machine vision and medical endoscopy [104].

Figures 5.3 and 5.4 show the system setup for this research as well as a top-down view of the UltraZed-EG FPGA board with the necessary connections for FPGA development.

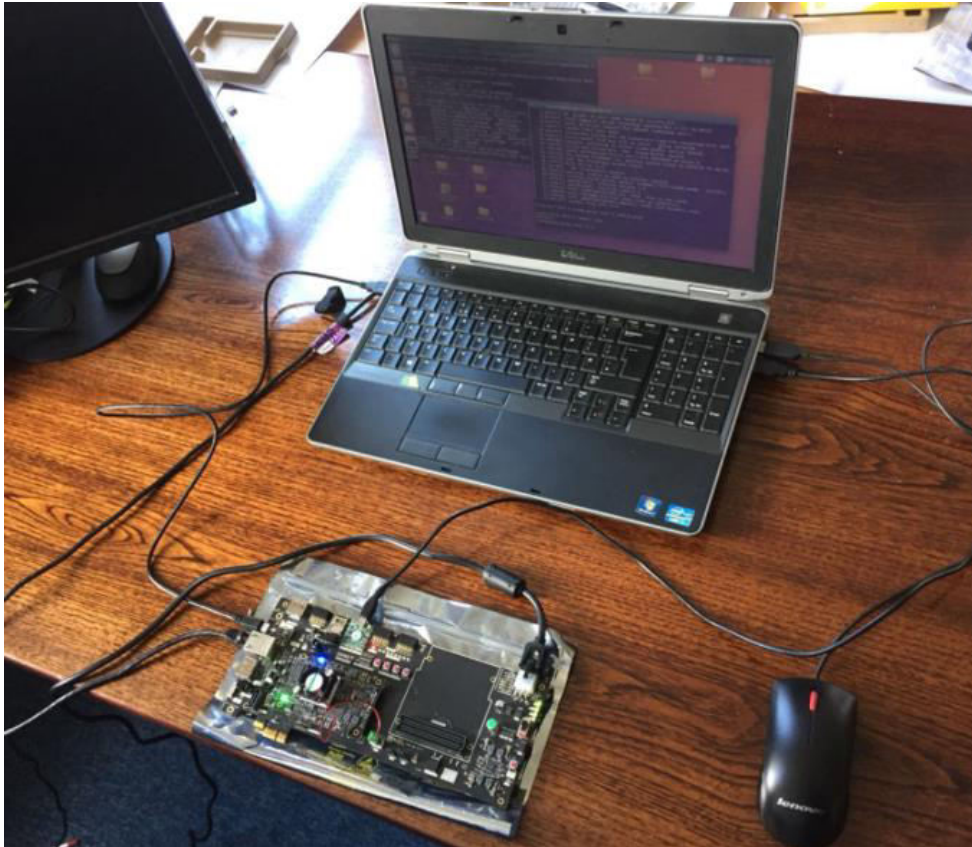


Figure 5.3: System setup for FPGA development.

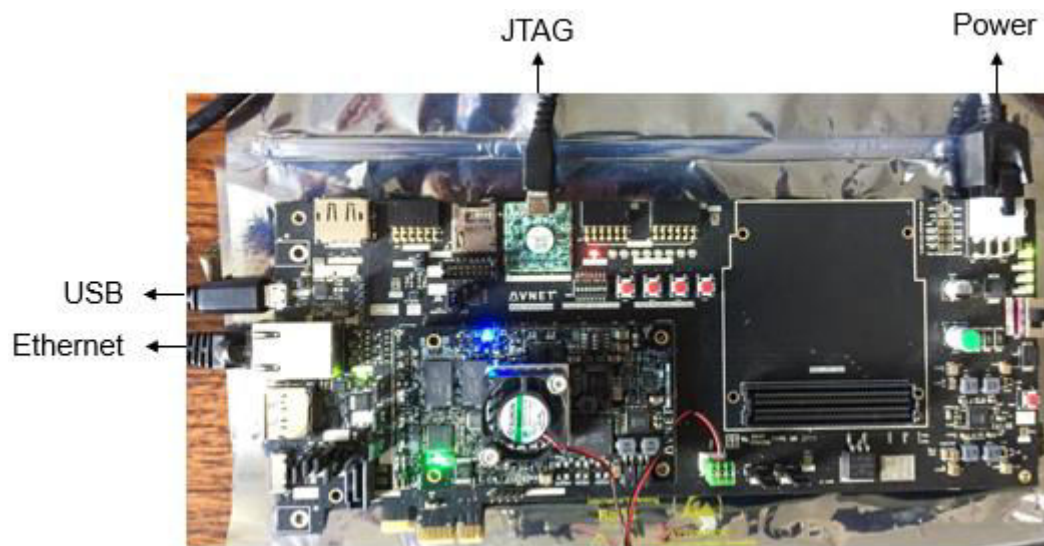


Figure 5.4: Xilinx UltraZed-EG board setup for FPGA development.

5.3. Development environment

This research makes extensive use of two libraries: OpenCV [105] and PCL [106].

The LiDAR pre-processing and navigation algorithms, as well as the point cloud generation from stereo images, have been developed using the PCL library and image pre-processing algorithms have been developed using the OpenCV library.

Once developed on the desktop computer, the image processing applications have been adapted for the FPGA board using the Xilinx Software Development Kit (SDK) Integrated Development Environment (IDE) [107]. The operating system on-board the FPGA board is PetaLinux [108] [109]. Finally, the serial communications program used to communicate between the FPGA board and the desktop computer during FPGA development is Minicom [110] [111].

5.3.1. OpenCV libraries

OpenCV is a computer vision and machine learning software library. With many contributors continuously updating and improving the library, it has over 2500 optimised algorithms for typical machine learning and computer vision applications, including object recognition, stereo processing operations and 2-D-to-3-D reconstructions among many others [105].

It is widely used by research organisations, companies and governmental bodies. Some examples of its users are well-established companies such as Google, Honda, Yahoo, Intel, Microsoft, IBM, Sony and Toyota, as well as startups like Applied Minds, Zeitera and VideoSurf [105].

OpenCV is written natively in C++ and has Python, Java and MATLAB interfaces. It supports Windows, Linux, Android and Mac operating systems [105].

5.3.2. Point Cloud Libraries (PCL)

PCL was born as a response to the increasing commercialisation of 3-D devices such as Kinect and the growing popularity of 3-D vision. As with OpenCV, it is also an open-source software library. It includes functions for filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation [106].

Natively written in C++, PCL is cross-platform and has been successfully implemented on Linux, MacOS, Windows, and Android/Ios. With the aim of distributing PCL on platforms with constrained size or computational power, PCL is modularised into a set of smaller libraries that can be compiled individually [112]. Figure 5.5 shows the relationships between the different modular libraries that PCL comprises.

PCL has a series of dependencies (3rd part libraries) that need to be installed prior to PCL's installation. They are [113]:

- **Boost** library: used for shared pointers, and threading.
- **Eigen** library: used as the matrix backend for SSE optimized math.
- **FLANN** library: used in the *kdtree* module for fast approximate nearest neighbours search.
- **Visualization ToolKit (VTK)** library: used in the *visualization* module for 3-D point cloud rendering and visualization.
- In addition, there are the optional libraries: googletest, QHULL, OpenNI and Qt.

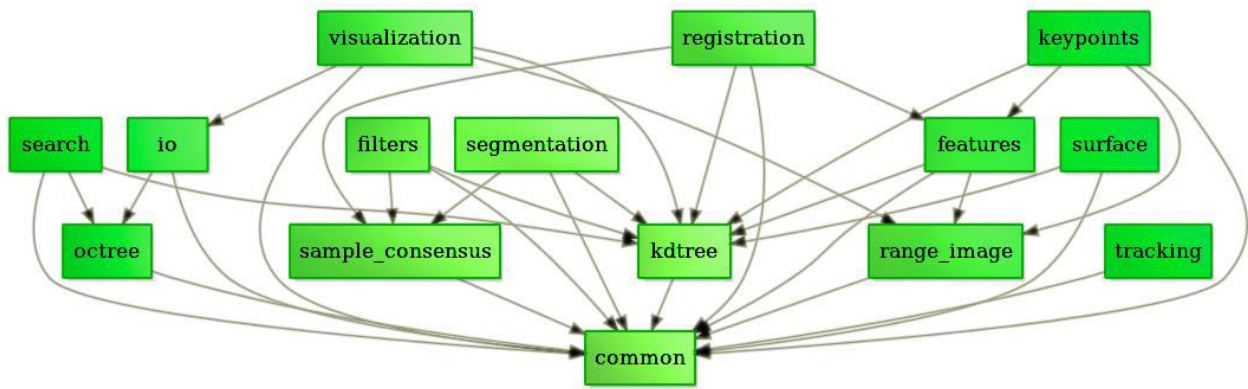


Figure 5.5: PCL libraries and the interconnections between them [112].

5.3.3. PetaLinux

PetaLinux is an Embedded Linux System Development Kit specifically developed for FPGA-based SoCs [108]. It offers all the necessary tools to develop Embedded Linux applications for Xilinx boards. When an application is developed using PetaLinux Tools, it will automatically create a custom Linux Board Support Package (BSP). A BSP includes all drivers for Xilinx embedded processing platforms, kernel and boot loader configurations. Once the applications have been built, PetaLinux Tools allows the user to package them appropriately for installation on the FPGA board and distribution [109]

The main advantage of using this Xilinx-specific package is that it is aware of Xilinx hardware development tools and custom-hardware-specific data files, which highly automates and eases the development process [109].

5.3.4. Xilinx Software Development Kit (XSDK)

XSDK is the Integrated Design Environment (IDE) to create applications for Xilinx processing boards. In [107] Xilinx claims that “SDK is the first application IDE to deliver true homogenous and heterogeneous multi-processor design, debug, and performance analysis.”

Figure 5.6 shows a screenshot taken while developing image processing applications using Xilinx SDK.

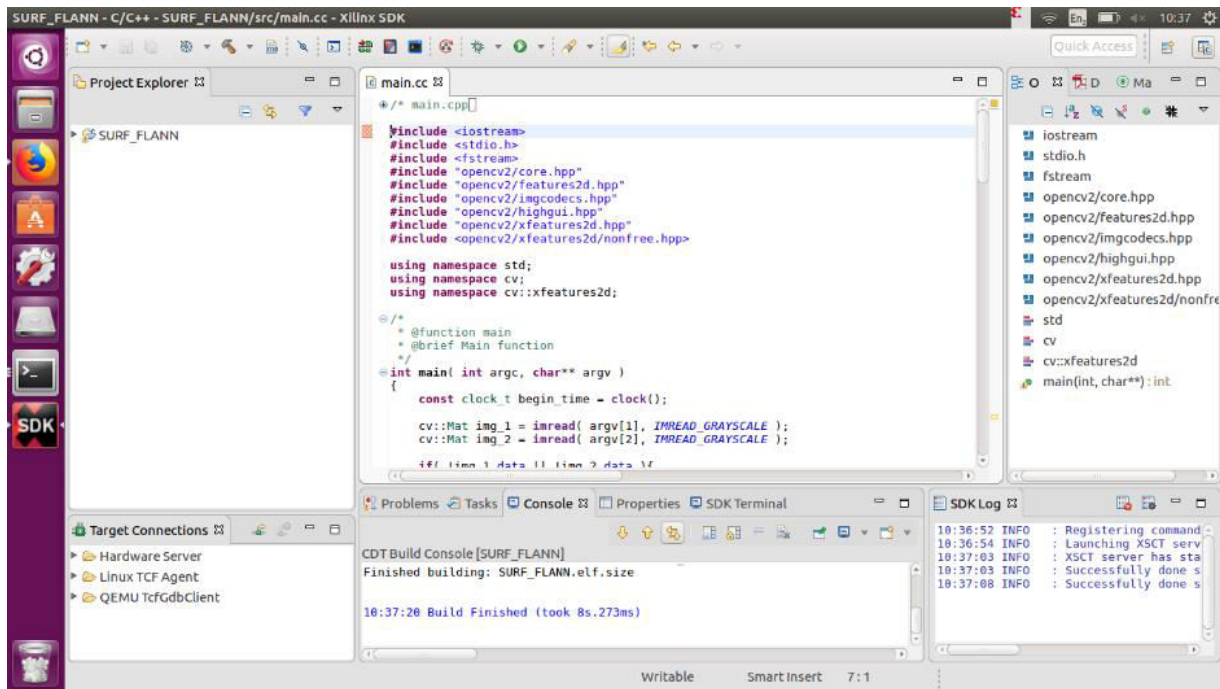


Figure 5.6: Screenshot of Xilinx SDK being used for the development of image processing applications.

5.3.5. Minicom

There is no specific requirement for the serial communications program to use for communication between the desktop computer (host computer) and the FPGA board. In this work the Minicom software program has been used for being one of the standards for Linux, which is the operating system installed on the host computer [110] [111].

Figure 5.7 shows an example of use of the Minicom program through the Linux console. In the image, the FPGA board has just been booted and accessed by inputting the login credentials.

```
labuser@labuser-Latitude-E6530: ~
[ 6.582591] usb usb2: Product: xHCI Host Controller
[ 6.587445] usb usb2: Manufacturer: Linux 4.9.0-xilinx-v2017.4 xhci-hcd
[ 6.594042] usb usb2: SerialNumber: xhci-hcd.0.auto
[ 6.599183] hub 2-0:1.0: USB hub found
[ 6.602882] hub 2-0:1.0: 1 port detected
[ 6.607577] OF: graph: no port node found in /xilinx_drm
[ 6.612824] [drm] Supports vblank timestamp caching Rev 2 (21.10.2013).
[ 6.619416] [drm] No driver support for vblank timestamp query.
[ 6.704477] random: fast init done
[10.689783] xilinx-drm xilinx_drm: No connectors reported connected with mods
[10.696942] [drm] Cannot find any crtc or sizes - going 1024x768
[10.709508] Console: switching to colour frame buffer device 128x48
[10.720742] xilinx-drm xilinx_drm: fb0: frame buffer device
[10.740750] [drm] Initialized xilinx_drm 1.0.0 20130509 on minor 0
[10.747140] rtc_zynqmp ffa60000.rtc: setting system clock to 1970-01-01 00:00
[10.755518] clk: Not disabling unused clocks
[10.759785] ALSA device list:
[10.762701] #0: DisplayPort monitor
[10.766411] Warning: unable to open an initial console.
[10.796529] Freeing unused kernel memory: 61824K (fffffc000c30000 - ffffffc)
[11.485573] udevd[1657]: starting version 3.2
[11.510674] udevd[1658]: starting eudev-3.2
[12.192932] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
[13.229050] macb ffe00000.ethernet eth0: link up (100/Full)
[13.234640] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[13.405267] random: crng init done

PetaLinux 2017.4 uz3eg-pciec-2017-2 /dev/ttyPS0

uz3eg-pciec-2017-2 login: root
Password:
root@uz3eg-pciec-2017-2:~#
```

Figure 5.7: FPGA login screen from Minicom serial port communication program.

5.4. Test bench setup design

Within the project test bench setup, LiDAR pre-processing and navigation algorithms have been developed using a standard PCU (i.e. desktop computer).

In parallel, the Xilinx UltraZed EG FPGA board has been used for image pre-processing and image-based navigation. Figure 5.8 shows a block diagram of the system design for FPGA development. Figures 5.3 and 5.4 are photographs taken during the development of this work and represent the system setup for FPGA development.

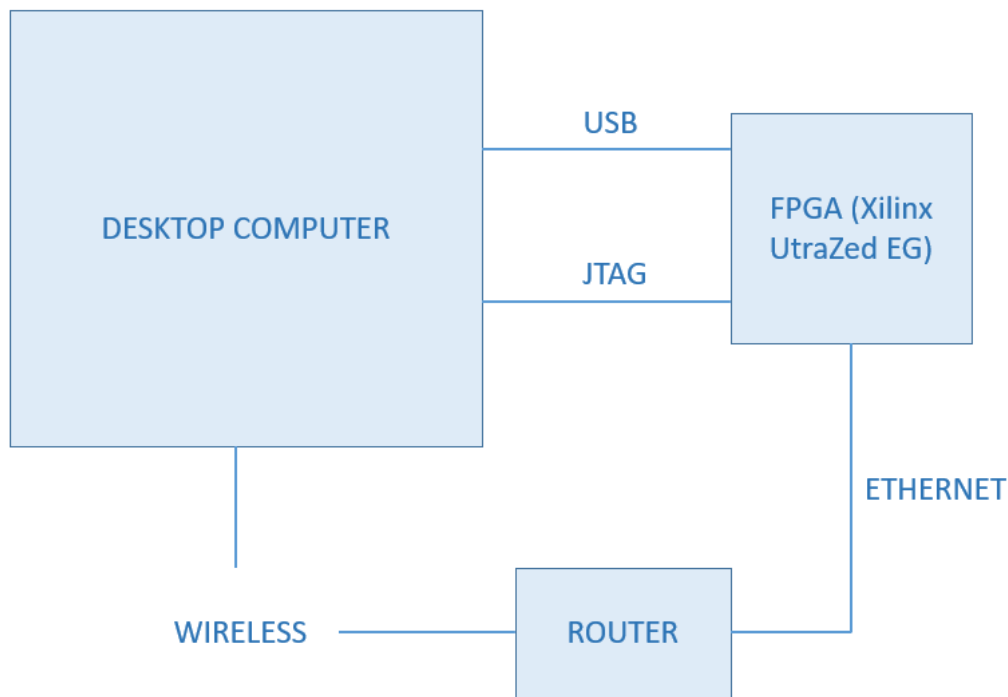


Figure 5.8: System design for FPGA development.

Once a system image is developed using PetaLinux Tools in a desktop computer, it is uploaded to the FPGA board through a JTAG (Joint Test Action Group) connection. At this moment the board is operational. A serial communication program (Minicom) is used to communicate, through USB, between the board and the desktop computer. Applications are developed and packaged to run on the board using the Xilinx SDK. Once the applications are created, they can be uploaded to the board through WiFi. A router, connected wirelessly to the computer and by an Ethernet cable to the FPGA board, establishes this connection.

The following steps must be followed during FPGA development to have the board operational, upload applications onto it and test them:

1. **PetaLinux working environment setup:** PetaLinux has some dependencies that must be installed to be able to use PetaLinux Tools. Besides these (specified in [108]), PetaLinux, Vivado and XSDK need to be downloaded from Xilinx website and installed. Thereafter, their “settings” scripts must be sourced.
2. **Create a Linux BSP (system image):** PetaLinux offers many libraries and functionalities that the user can include or not in a specific BSP, according to the different needs. This configuration, as well as building and packaging the BSP, can be done using PetaLinux commands [108].
3. **Boot the FPGA board with the created system image:** A PetaLinux image can be booted on hardware through JTAG (Joint Test Action Group), TFTP (Trivial File Transfer Protocol) or using a SD card [108]. Although both the SD card and JTAG methods were tested in this work, JTAG was preferred due to its utilization by the other I3DS partners.

Once the system image has been built and the FPGA board is connected to the desktop computer as shown in Figure 5.3 and switched on, having a minicom console and a terminal

with the PetaLinux environment appropriately on the host computer, the image can be booted using PetaLinux commands on this second terminal.

After about 5 minutes the FPGA board is operational and a login screen like the one on Figure 5.7 appears on the Minicom console.

4. **Develop applications using XSDK:** To develop applications on Xilinx SDK it is not necessary to have the FPGA operational, but it is recommended in order to test applications on hardware once they are developed. XSDK is like any other IDE, very similar to Eclipse, and minor modifications must be done to Linux program codes to be transferred to the FPGA board. For example, the versions of OpenCV used in the desktop computer and on the FPGA board are different and thus some of the functions in the image processing codes must be modified. Additionally, not all libraries are available in PetaLinux, which also requires changing some of the functions used in the codes. Finally, because there is no display option on the board, any image visualisation functionalities must be omitted.
5. **Transfer files between the host computer and the FPGA board:** To be able to test the image processing algorithms on the FPGA board, the test images must be first stored on it. This can be done by transferring them from the host computer via the Internet (being the connection between the FPGA-router through Ethernet and host computer-router through wireless), using SFTP (SSH File Transfer Protocol) protocol. This connection between the FPGA board and the host computer is established on a separate console on the host computer. Although it is necessary to store the data on the FPGA board during FPGA development and testing, it is not expected that during normal operation in the I3DS project, when the INSES is installed on a satellite, sensor data are stored on-board. Instead they will be directly processed “on-the-fly” and discarded.
6. **Upload and test the applications onto the FPGA board:** Once an application has been developed on XSDK, it is uploaded onto the FPGA board via the Internet connection (Ethernet between the board and the router and wireless between the router and the host computer). This is done using the SSH (Secure Shell) protocol. The executable file is then directly created on the board. Further run commands, such as the directory on the board of the data to be processed, can be specified using XSDK. XSDK has a console that shows the result of the operations, coming from the board. Alternatively, once the executable is created on the FPGA board it can be run directly in the Minicom console.

5.5. LiDAR processing results

Because of the high processing power needed to process point clouds and because the library used to process them is PCL, which is not available for the ARM architecture on the FPGA board, the end-to-end LiDAR processing will be implemented on the I3DS OBC. As such, the results presented here have been obtained with the desktop computer simulating the OBC.

5.5.1. Point cloud pre-processing

Different point cloud down-sampling rates have been evaluated with the aim to select the best trade-off between computing time and navigation accuracy. The results showed that the best solution was achieved when the point clouds were down-sampled to sizes ranging from 100 to 800 points.

The uniform sampling filter used in this work divides the point cloud in voxels of a certain radius, here called “down-sampling radius”, and approximates each voxel with its centroid. The filter takes as input the down-sampling radius. This radius is a trade-off between computing time and the posterior navigation accuracy. A larger radius will result in an output point cloud with fewer points, which will reduce the computing time. At the same time, fewer points will also lead to a generally lower navigation accuracy. Furthermore, the ICP algorithm and its variants need a minimum number of points in order to be able to perform registration between two point clouds.

However, the size of the point clouds varies as the chaser satellite approaches the target satellite, i.e. when the two satellites are close part of the target satellite leaves the FOV of the LiDAR sensor and the size of the sensed point cloud is therefore reduced. Indeed, tests highlighted that using a constant down-sampling radius results in poor performance. A too large radius will cause that when the chaser satellite gets close to the target satellite and the size of the point cloud diminishes there are not enough points to perform an accurate registration. On the contrary, if the radius is too small the output point cloud computed when the two satellites are far will have many points and the computing time will be too high for real-time applications.

Thus, a modification was made on the algorithm: Instead of inputting a fixed down-sampling radius, the inputs were an upper boundary and a lower boundary for the number of points in the output point cloud. In this way, the computing time could be limited while ensuring a sufficient number of points for point cloud registration.

Figure 5.9 show the computing time and frequency of both the down-sampling and outlier removal operations, as a function of the output number of points for the uniform sampling filter and the input number of points for the statistical outlier removal filter. It must be noted that the output point cloud of the uniform sampling filter is the input point cloud of the outlier removal filter. Furthermore, whereas the computing time for the uniform sampling filter decreases with a higher number of points in the output point cloud, that of the posterior outlier removal filter increases with a higher number of points in its input point cloud. Therefore, the choice of the boundaries for the number of points resulting from the uniform sampling must take this into consideration. However, because the computing time of the uniform sampling filter is one order on magnitude larger than that of the outlier removal filter, this first operation should be given more weight in this decision.

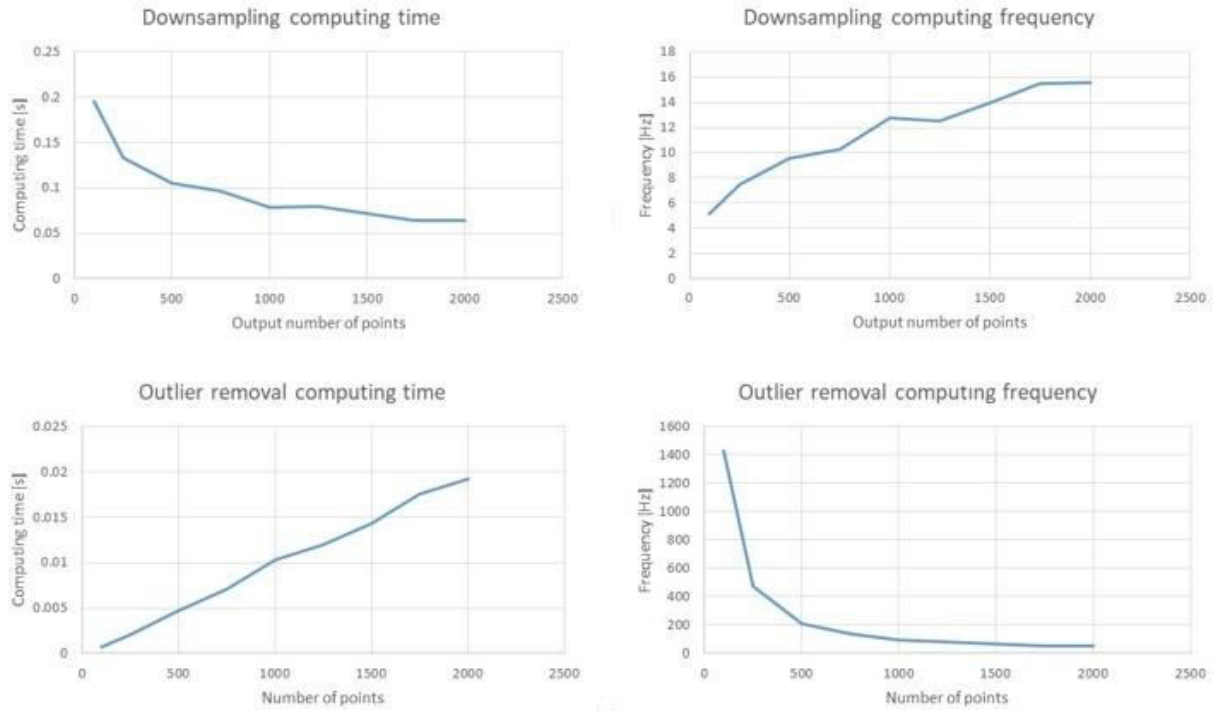


Figure 5.9: Computing time for point cloud pre-processing algorithms, for a raw point cloud size of 152100 points (5 MB).

5.5.2. Point cloud-based navigation

The algorithms presented in this section could be used for scan-based or image-based point clouds. Within this project the latter could result from the stereo images and be integrated with LiDAR data for a fusion-based navigation. However, this is out of the scope of this thesis and therefore, the results showed here correspond solely to LiDAR-based navigation.

Three different variants of the ICP point cloud registration algorithm were assessed in this work, namely, the classical ICP, GICP (Generalised ICP) and TrICP (Trimmed ICP). The performance of ICP and GICP depend on the initial transformation matrix. PCL allows the user to specify an initial guess for the transformation matrix for ICP and GICP and initializes it as the identity matrix if no initial guess is provided. Both approaches, i.e. considering an initial guess and with no initial guess, were evaluated. Because the trajectory is a straight-line approach at a constant speed and acquisition rate, an initial guess equal to the transformation matrix between the two previous frames was considered a good initial guess. TrICP allows the specification of an overlap ratio. In this implementation, ratios of 70%, 80% and 90% were evaluated.

The results of the navigation algorithms are expressed in the LiDAR reference frame at the initial point of its trajectory. In these axes, the straight-line results as represented in Figure 5.10, that depicts both the ground truth (blue) and the trajectory obtained with Generalised ICP (red). As can be seen, in these axes the chaser satellite advances monotonically in a straight-line from 0 m to 0.35 m in the x-axis, from 0 m to -0.13 m in the y-axis and from 0 m to 18.50 m in the z-axis. Therefore, most of the movement is in the z-axis. It is important to consider that in the last position of the trajectory, contact is established with the target satellite. Out of 200 point clouds that represent the trajectory, this position is reached in point cloud number 187, after which the chaser satellite is still.

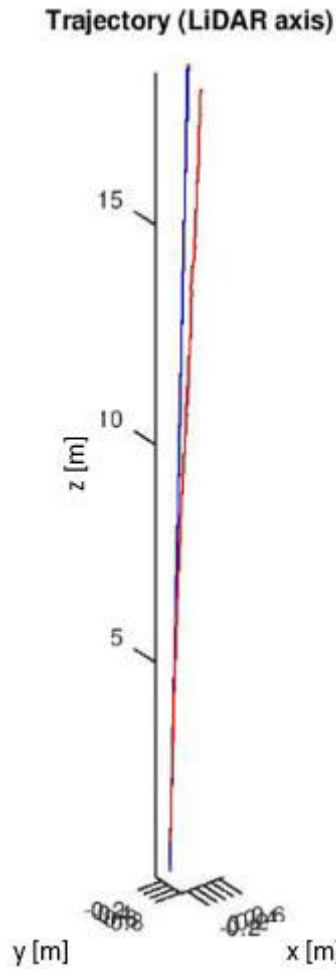


Figure 5.10: Ground truth (blue) and LiDAR-reconstructed trajectory (red) for straight-line approach mission.

Figures 5.11 and 5.12 show the errors acquired for the simulated trajectory in the three axes as a function of the frames (i.e. point cloud number) as well as of the covered distance, for all ICP variants. These errors are expressed in metres. Finally, Figure 5.13 shows the square error expressed in metres and percentage and both as a function of the frames and of the covered distance.

Figure 5.11 shows (red dashed line) that from point cloud 185, that corresponds with the moment in the trajectory when contact between the chaser spacecraft and the target spacecraft is made, there is a peak in all errors. This is because at this point the point cloud sensed by the LiDAR is too small to allow for a good registration. However, other I3DS sensors have been designed for this moment of the space rendezvous, namely, the force-torque sensor and the tactile sensors.

The square error in the trajectory determination is always below 3.5 %. Furthermore, the best performance for this particular scenario is achieved with the GICP algorithm and without an initial guess. Therefore, this is the algorithm recommended for the final approach to a target satellite in a space rendezvous.

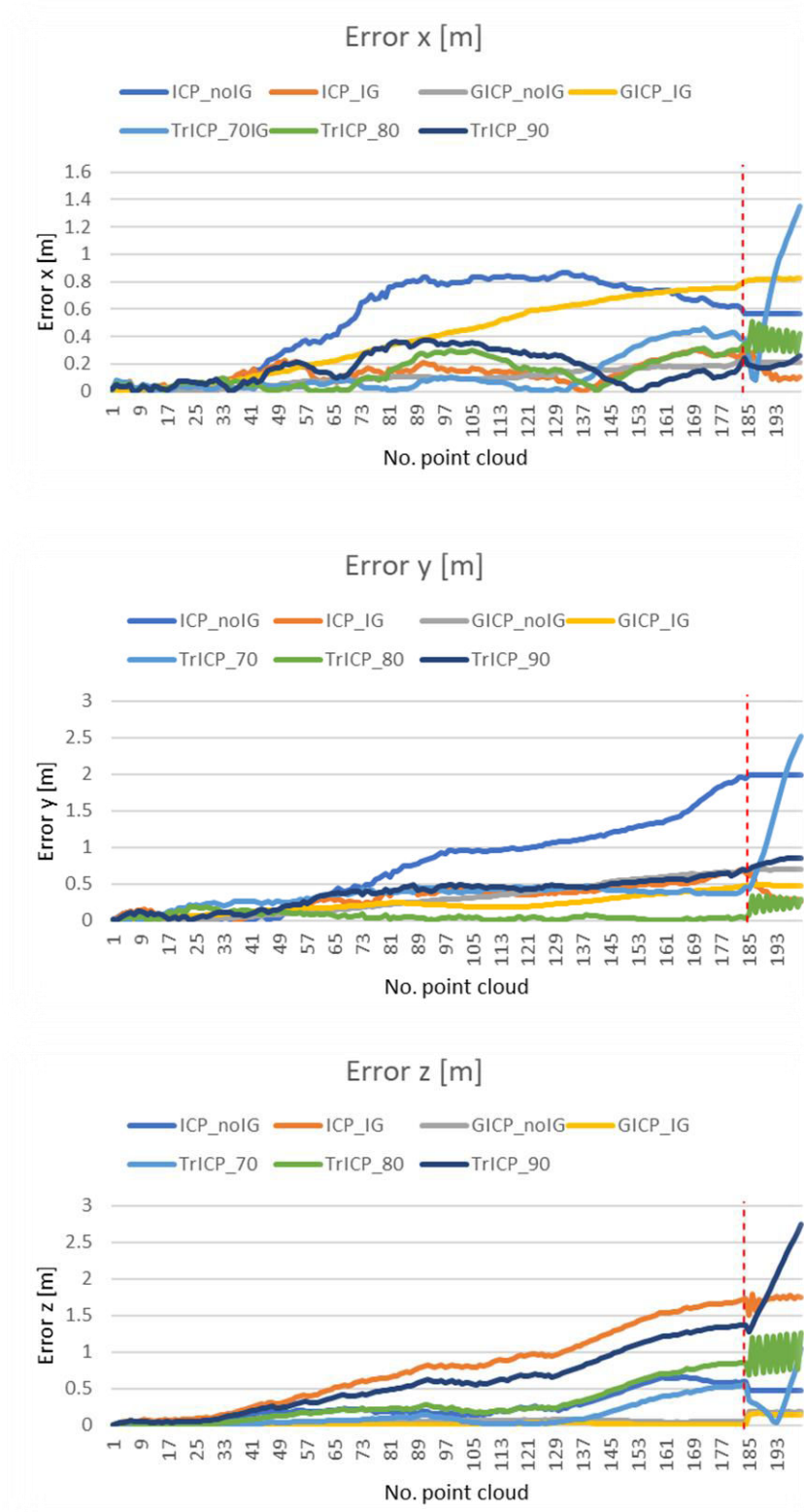


Figure 5.11: Errors in the x , y and z -axis, expressed in metres. For ICP and GICP, using or not an initial guess (IG) is considered. For TrICP, overlap ratios of 70%, 80% and 90% are evaluated. The results are expressed as a function of the frame / point cloud number.

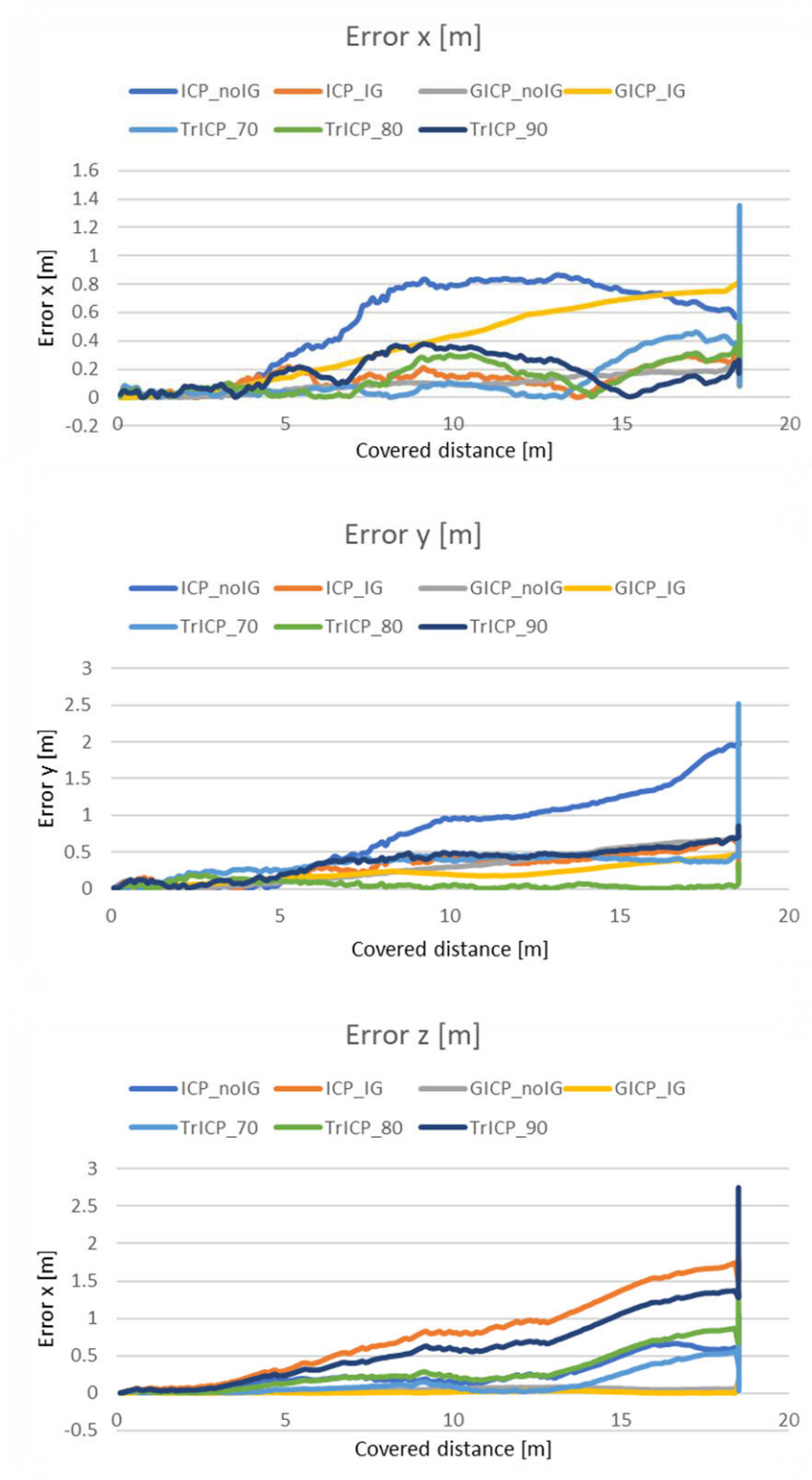


Figure 5.12: Errors in the x , y and z -axis, expressed in metres. For ICP and GICP, using or not an initial guess (IG) is considered. For TrICP, overlap ratios of 70%, 80% and 90% are evaluated. The results are expressed as a function of the covered distance.

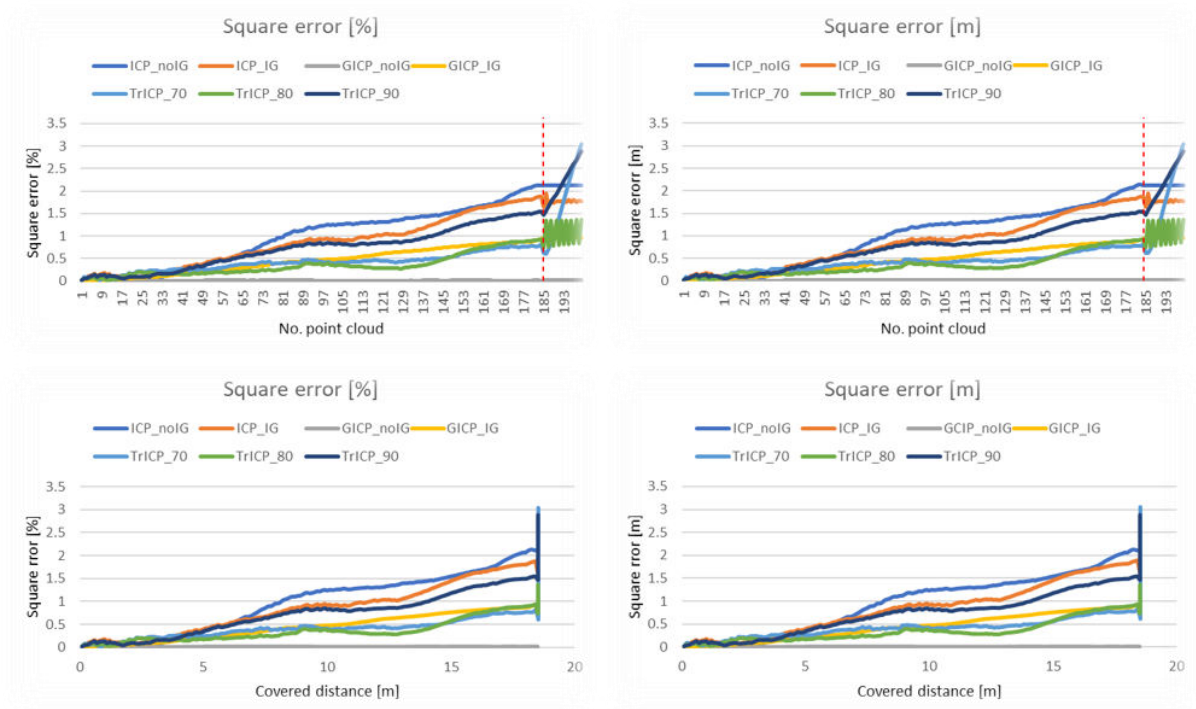
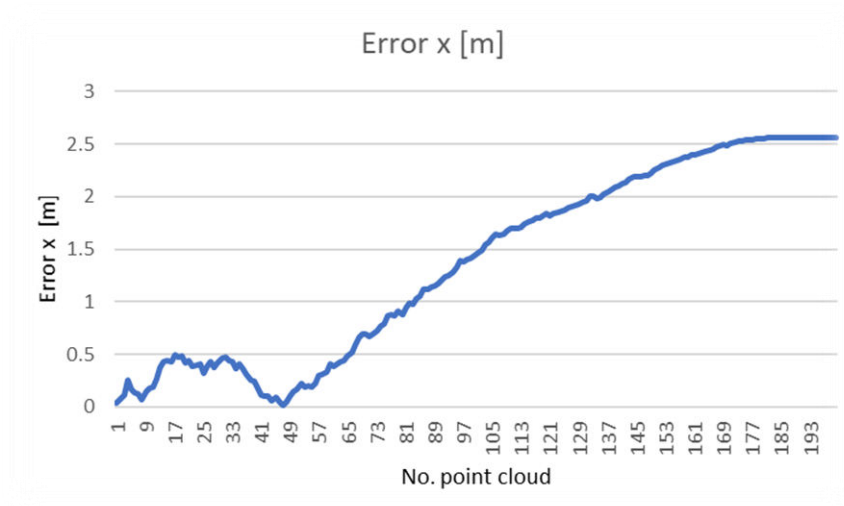


Figure 5.13: Square error in percentage and in meters in function of the point cloud number and of the covered distance for all the ICP variants.

Figures 5.14 to 5.16 show the results achieved with the HoD point descriptor and Kalman filter-based registration. Results on the simulated data show comparatively poor performance, with a mostly monotonically increasing error reaching 80%. Consequently, this algorithm was considered to be not suitable in the present form for this particular application.

Although further analysis with other data sets and other trajectories would be necessary, the reason for the “HoD+Kalman” navigation filter failing in this case is believed to be due to the morphology of the spacecraft. The HoD point descriptor characterises each point by its distances to its neighbours. Therefore, uniform surfaces such as the planes comprising a spacecraft will likely not achieve good results, since the distances between points are highly similar. On the contrary, this descriptor could possibly achieve better performance with more complex surfaces. For example, it might be of use for planetary exploration.



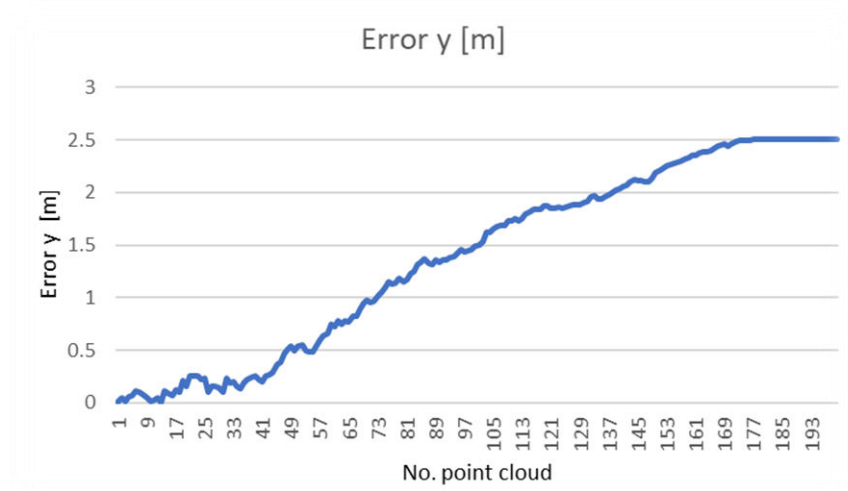
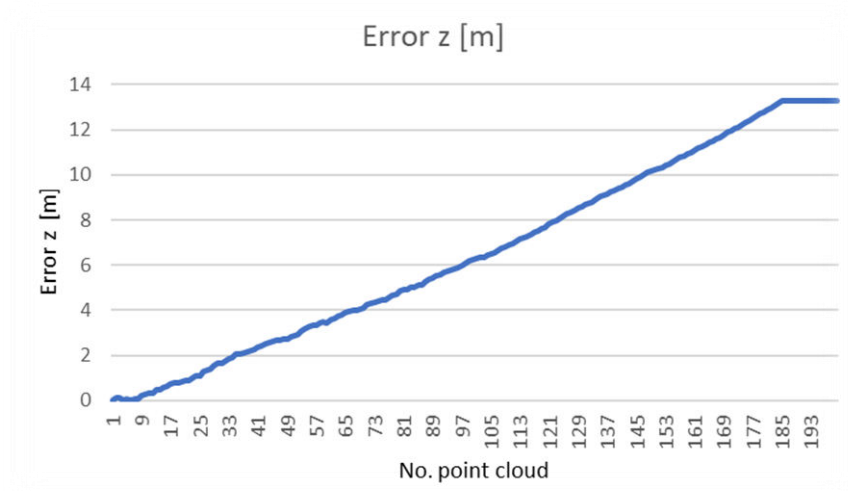
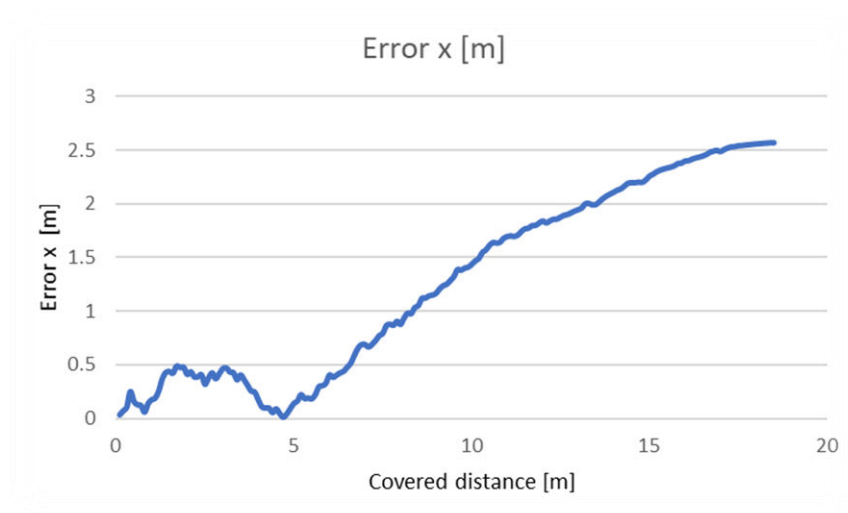


Figure 5.14: Errors in the x, y and z-axis, expressed in metres for HoD+Kalman navigation. The results are expressed as a function of the frame / point cloud number.



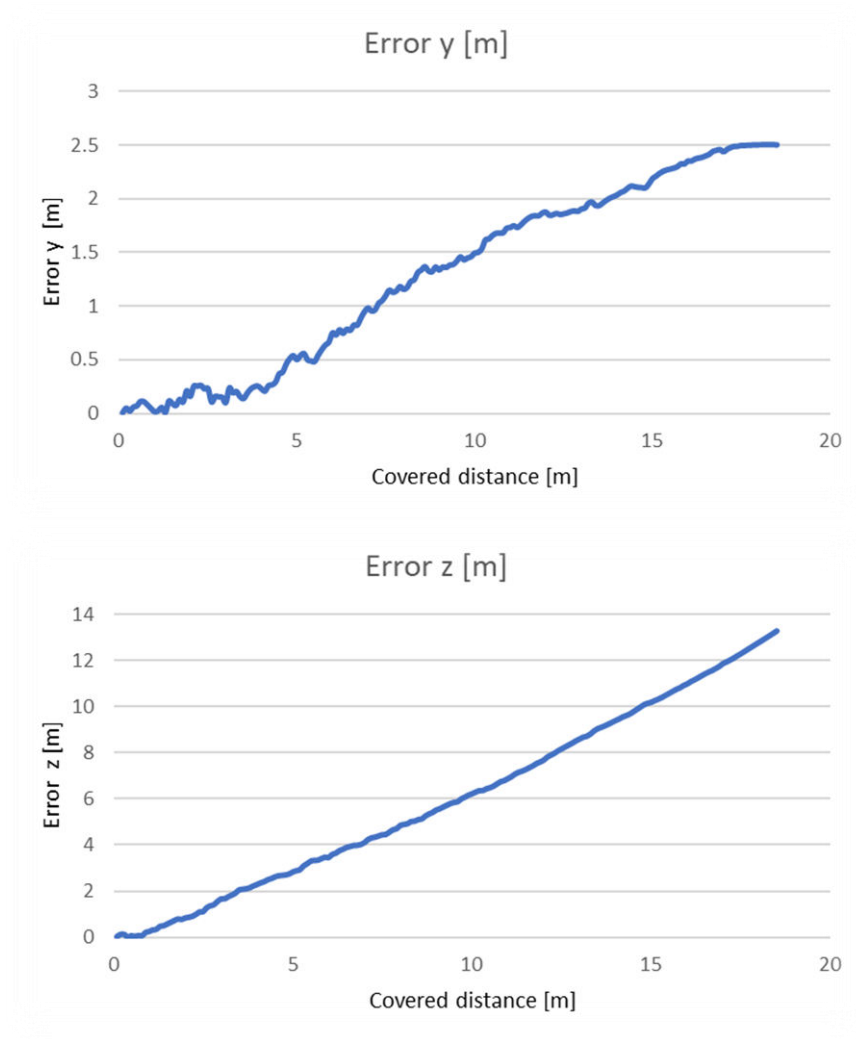


Figure 5.15: Errors in the x, y and z-axis, expressed in metres for HoD+Kalman navigation. The results are expressed as a function of the covered distance.

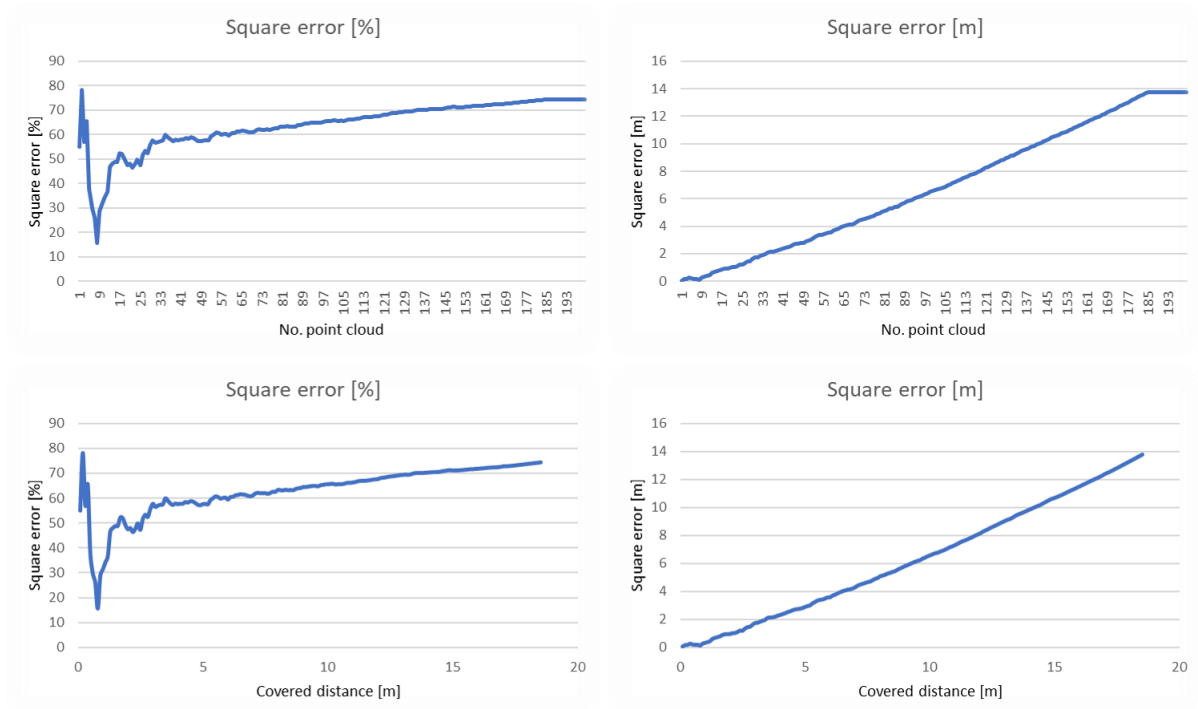


Figure 5.16: Square error in percentage and in meters in function of the point cloud number and of the covered distance for HoD+Kalman navigation.

Finally, Table 5.2 shows the average computing times for all of the evaluated LiDAR navigation algorithms. For the simulated scenario, the computing time per frame of all ICP variants is shorter than that of the HoD+Kalman algorithm. TrICP is the fastest algorithm, which might be caused to the method not trying to find a correspondence for all points in the point cloud. Out of the ICP variants, GICP is the most computationally expensive, especially when an initial guess is not provided. However, the acquisition frequency of the LiDAR sensor simulated by TAS-F was 0.5 fps, which results in a computing time requirement of 2 seconds per frame. Therefore, even with the 0.57 seconds for GICP this requirement is met.

Table 5.2: Average computing time for the studied point cloud-based navigation algorithms.

Navigation algorithm	Mean computing time per frame
ICP without an initial guess	0.03 s
ICP with an initial guess	0.03 s
GICP without an initial guess	0.57 s
GICP with an initial guess	0.32 s
TrICP with a 70% overlap	0.01 s
TrICP with an 80% overlap	0.01 s
TrICP with a 90% overlap	0.01 s
HoD + Kalman	2.41 s

5.6. Image processing results

This section presents the results achieved for image processing on both the desktop computer and the FPGA. The aim of this comparison is to assess the capabilities of FPGA technology to perform part of the on-board processing operations traditionally carried out on the OBC.

Because the accuracy of the navigation algorithms does not depend on the computing hardware but on the algorithms used, the comparison between the desktop computer and the FPGA board will be limited to the computing time only.

It must be noted that the original I3DS-defined requirement for end-to-end image processing is 10 fps.

5.6.1. Image pre-processing

Figures 5.17 and 5.18 show the computing time of the three image pre-processing operations involved, i.e. lens distortion correction, histogram equalisation and bilateral filtering.

Histogram equalisation is the fastest algorithm on both the desktop computer and the FPGA. However, the most time-consuming one on the desktop computer is lens distortion correction, whereas the most time-consuming algorithm on the FPGA board is bilateral filtering. Furthermore, the computational time between bilateral filtering and lens distortion correction is considerably larger for the FPGA board than for the desktop computer. For example, for a 0.5 MB image the FPGA board is able to run bilateral filtering in 39 ms and lens distortion in 10.82 ms, while the desktop computer takes 5.44 ms for lens distortion and 4.5 ms for bilateral filtering. This difference may be due to the different versions of OpenCV implemented on both hardware platforms and to possible differences in optimisation for certain algorithms between versions.

The overall computing time, including the three algorithms, is 0.018 sec/frame on the desktop computer and 0.14 s/frame on the FPGA board.

As can be observed, the computing time on the desktop computer is considerably shorter than on the FPGA board. However, the use of an FPGA in the I3DS sensor suite is based on its capability to undertake some of the computing load of the OBC and so reduce the total computing time for multi-sensor navigation.

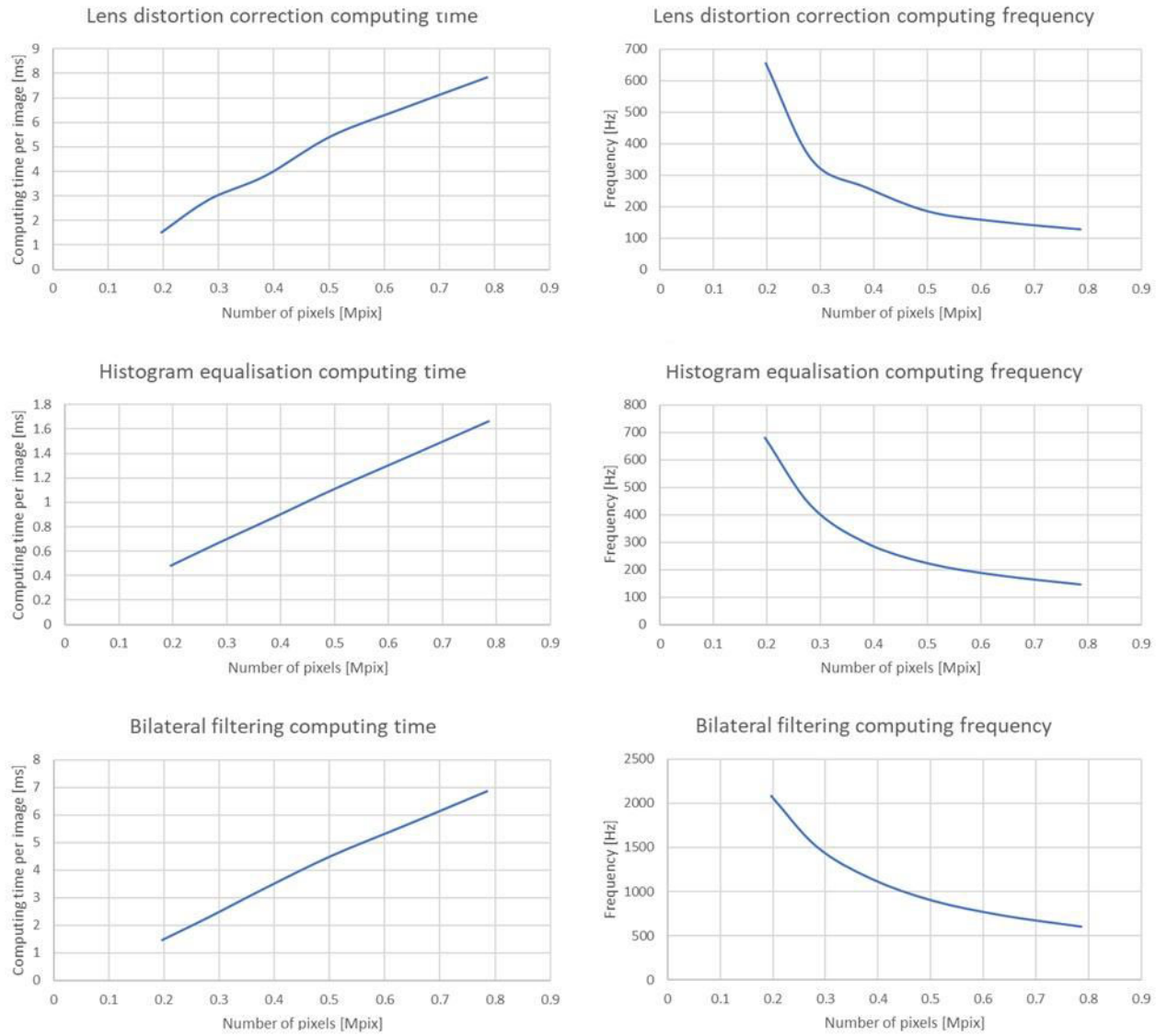


Figure 5.17: Computing time and frequency of image pre-processing algorithms on the desktop computer.

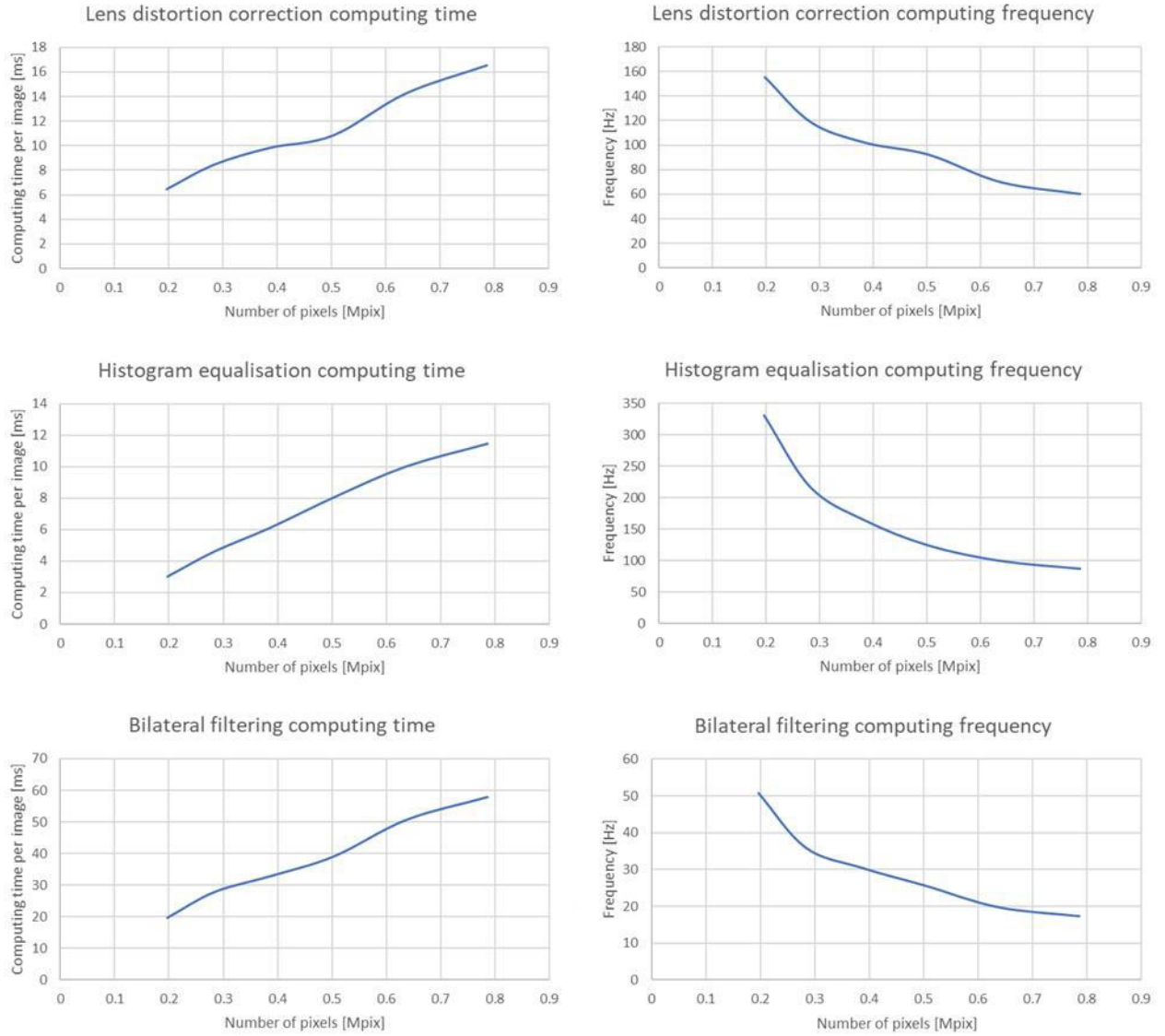


Figure 5.18: Computing time and frequency of image pre-processing algorithms on the FPGA board.

5.6.2. High-resolution-camera-based navigation

In the camera axes, the motion for the simulated trajectory is along the z-axis and the target z-axis is perfectly aligned with the camera z-axis, therefore no yaw, pitch and roll are present. Since for this kind of navigation filter the errors do not sum up, the plots shown will involve exclusively the errors for each frame.

A detailed error analysis is presented in Figure 5.19 for the trajectory and Figure 5.20 for the attitude. Throughout the entire mission the error in position of the target is almost constant and below 2 cm. At about 6 m, the 4 markers leave the FOV, causing a slight variation in the attitude estimate and position. This is natural, since the PnP is an optimization method and the more points detected and associated in the target, the better the estimate. The major effect of the missing markers is in the roll in this case. It is worth noting that misplacement of the 2-D position of the centre of the markers in the image affected mainly the roll and pitch angles, as shown in Figure 5.20. In general, it clearly depends on the position of the undetected markers.

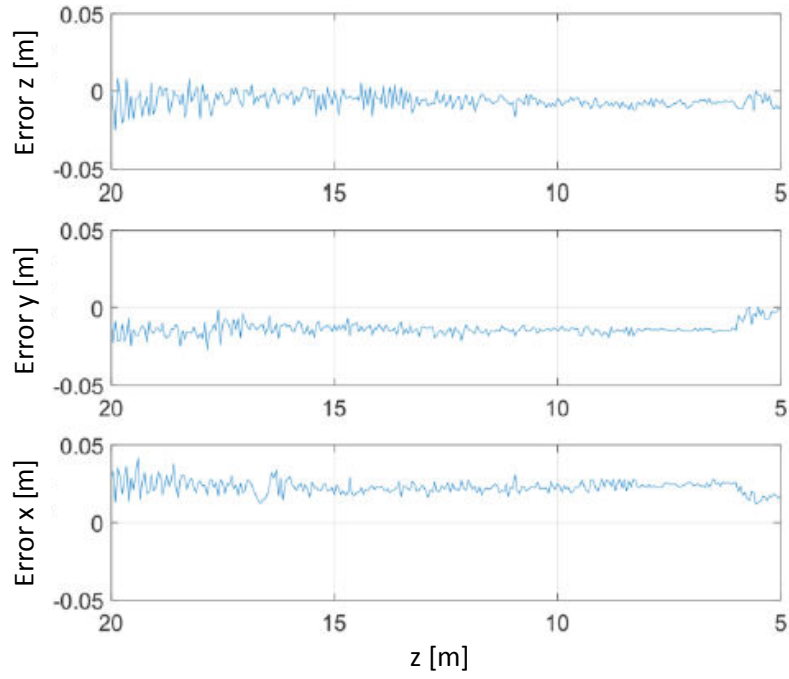


Figure 5.19: Trajectory errors for straight-line approach with markers-only filter.

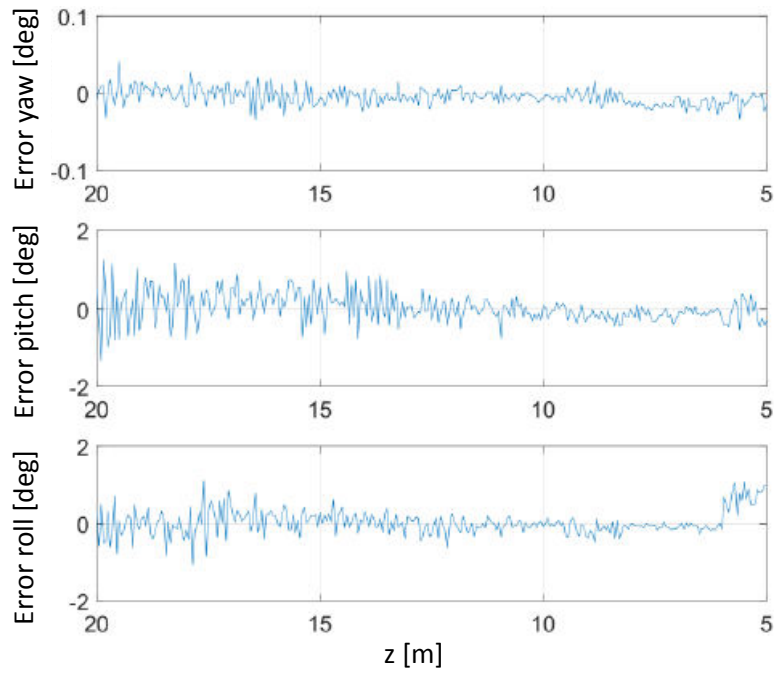


Figure 5.20: Attitude errors for straight-line approach with markers-only filter.

Due to the inter-frame displacement being comparable in magnitude to the error, a slower frame-rate was simulated. The mission profile is shown in Figure 5.21. The frame-rate is about 0.13 Hz.

The estimated trajectory, as shown in Figures 5.22 and 5.23 properly follows the ground truth trajectory. It is worth noting that reconstruction of the trajectory stops when the fiducial markers are not in view anymore.

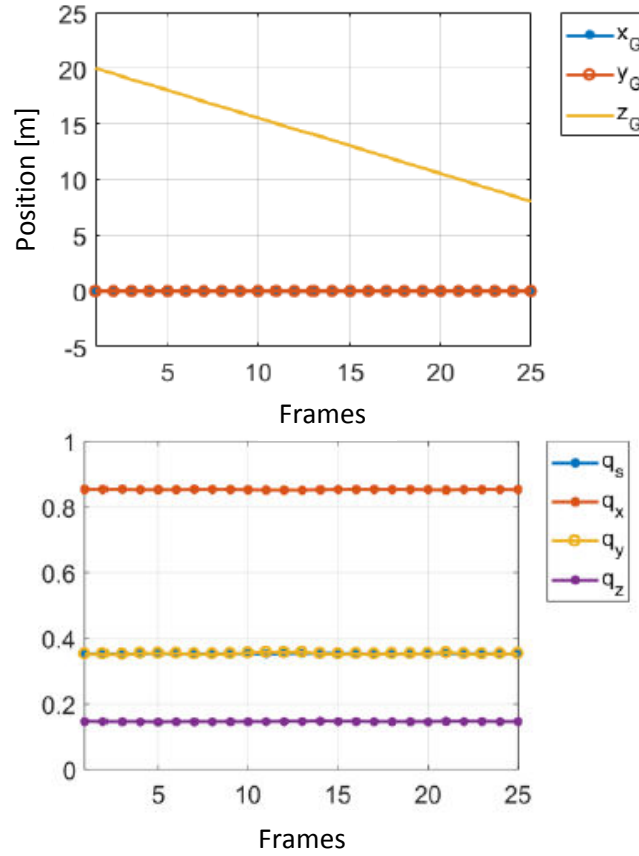


Figure 5.21: Reconstructed translation and rotation for the second navigation case.

5.6.3. Stereo-camera-based navigation

The first trajectory is the simulated straight-line from 20 m to 5 m with the chaser satellite approaching the target satellite at a speed of 5 cm/s. As for the HR camera, the motion is along the camera z-axis and the target z-axis is perfectly aligned with the camera z-axis, therefore no yaw, pitch and roll are present.

For the case of stereo-camera-based navigation, despite the solution being able to run at speeds greater than 10 Hz, a framerate of 10 Hz led to erroneous results. This is because at a speed of 5cm/s and framerate of 10 Hz, the inter-frame motion would be 5mm, which is too small for the proposed stereo-camera to track the feature motion properly. Indeed, the inter-frame disparity is too small and consequently the signal to noise ratio of the tracked features is insufficient to allow for filter convergence.

Therefore, different framerates have been tested, namely 1Hz and 2Hz.

Figures 5.22 and 5.23 show the estimated trajectory and the estimated attitude (in blue), respectively, together with the ground truth (in red). For this analysis it is more intuitive to express the attitude using the yaw, pitch, roll convention, instead of using quaternions as this will give a more thorough understanding of the results. Conversion to quaternions for control purposes is however straightforward. Overall, the estimated pose follows the ground truth trajectory, even though errors on the final position are present. It is therefore necessary to analyse the errors to have a better indication of the current performance and eventually improve the tuning.

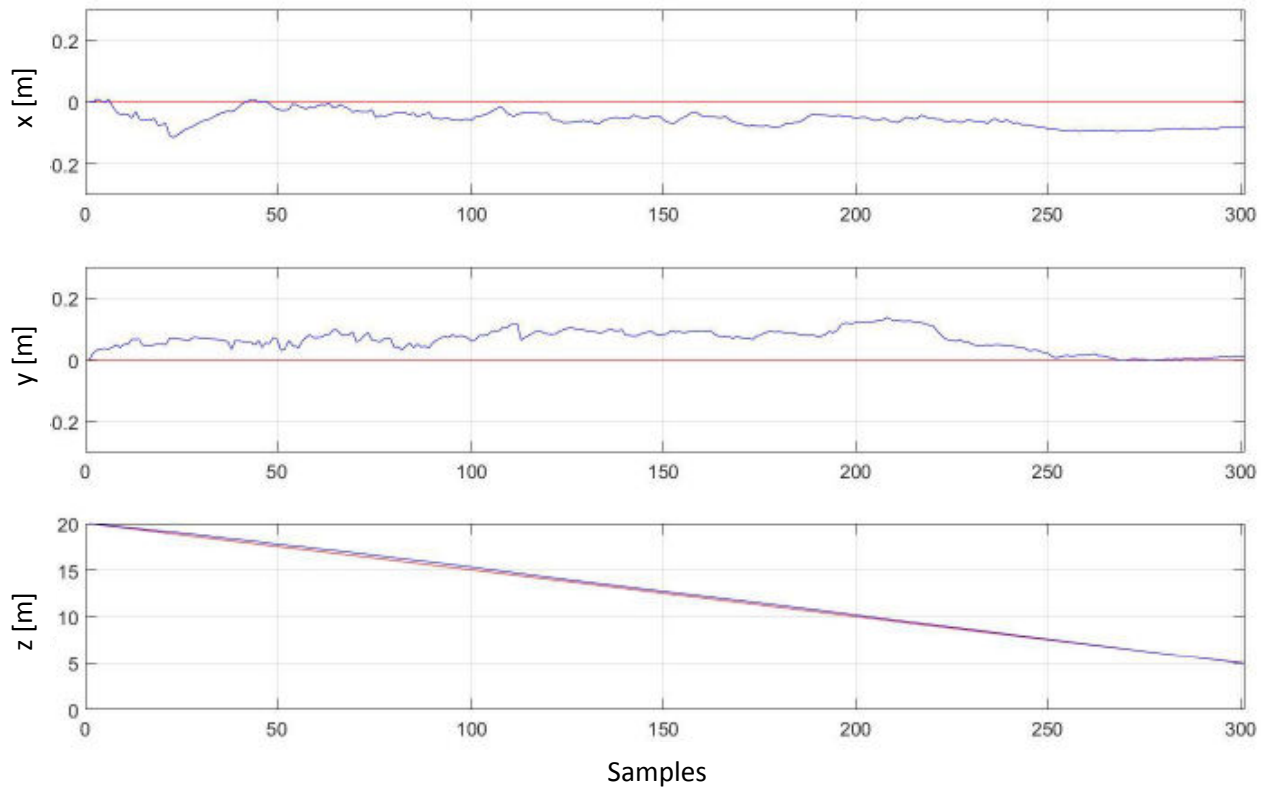


Figure 5.22: Estimated trajectory (red) and ground truth (blue) for straight-line approach with stereo navigation filter.

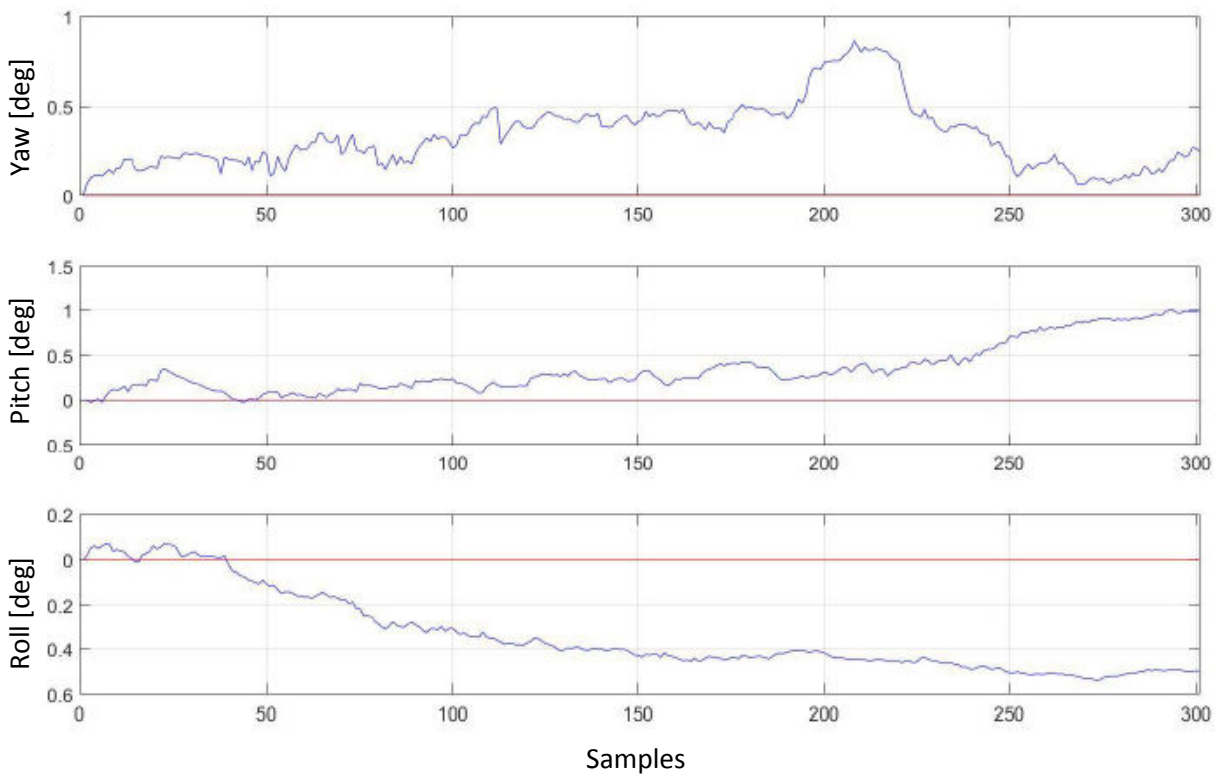


Figure 5.23: Estimated attitude (red) and ground truth (blue) for straight-line approach with stereo navigation filter.

The inter-frame motion errors are analysed first. The inter-frame motion error allows an understanding of the performance on a frame-by-frame level. Figure 5.24 shows the inter-frame errors for the trajectory. Larger inter-frame errors are present at the beginning of the trajectory,

when the platform is further away. This can be explained by the fact that the disparity, i.e. the difference in pixels along the baseline direction between corresponding points in left and right images, is very small and hinders the process. Indeed, since depth is inversely proportional to disparity, very slight errors in disparity due to image noise, when the overall disparity is very small, can lead to very large errors in depth. Additionally, also the difference in motion of the features between two successive frames at large distance is minimal and thus may affect the accuracy of the results. When the target gets closer, the algorithm tracks the attitude much better and the inter-frame error decreases to the sub-centimetre level. The plot in Figure 5.25 shows the inter-frame errors for attitude. In general, they show an almost constant trend, fluctuating around zero value.

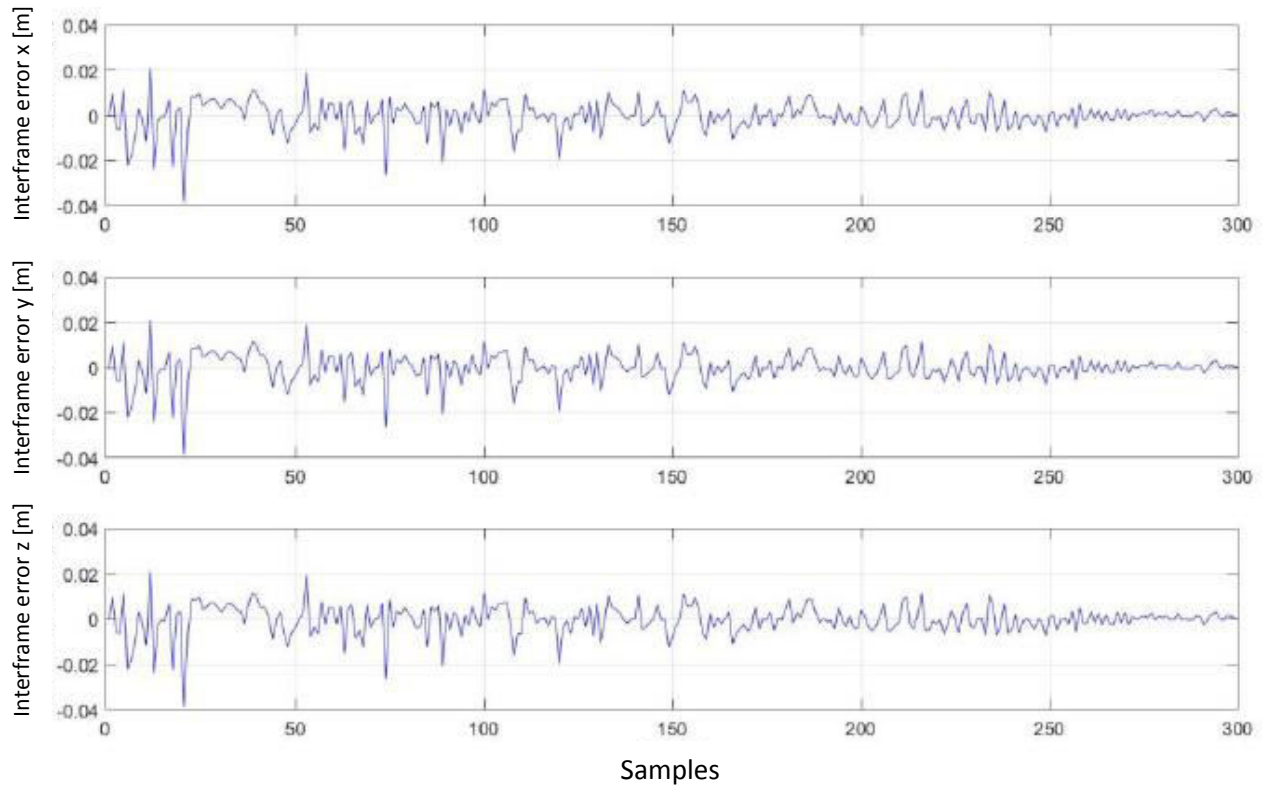


Figure 5.24: Inter-frame errors for trajectory in straight-line approach with stereo navigation filter.

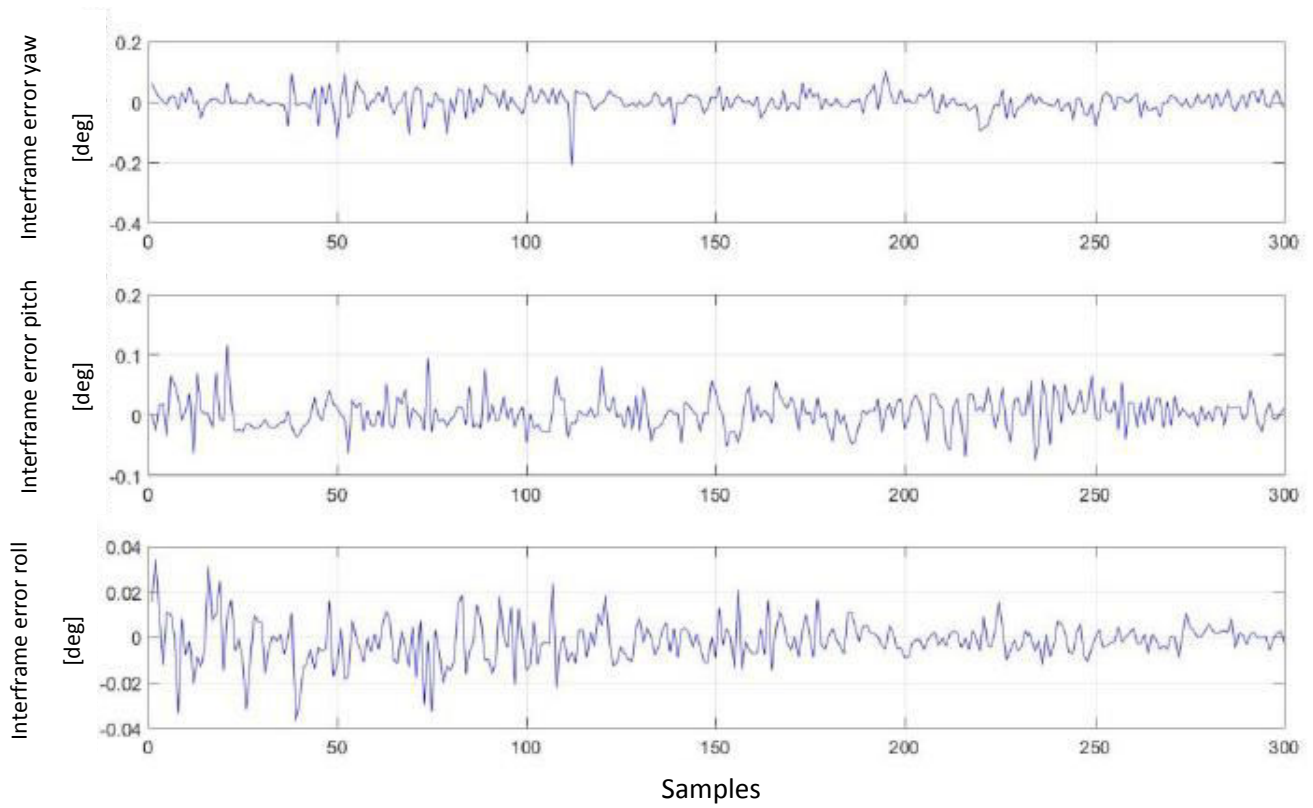


Figure 5.25: Inter-frame errors for attitude in straight-line approach with stereo navigation filter.

The sum of inter-frame motion errors gives the cumulative errors. Cumulative errors are plotted in Figure 5.26 for the trajectory and Figure 5.27 for the attitude. The final error in the trajectory is smaller than 2% of the entire path. However, the inter-frame errors accumulate and give fluctuations. Hence, further refinements should be tested to reduce these errors and perhaps also the effects of using different baselines.

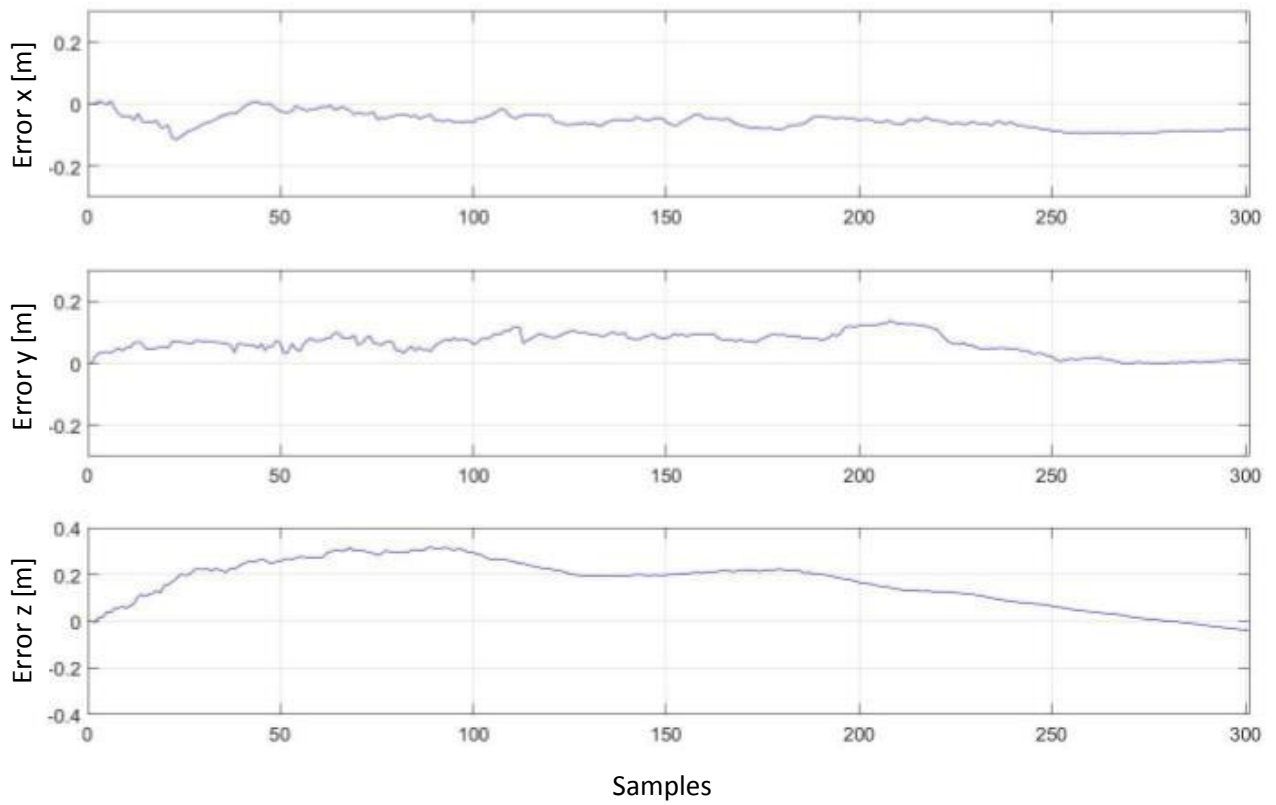


Figure 5.26: Cumulative errors for trajectory in straight-line approach with stereo navigation filter.

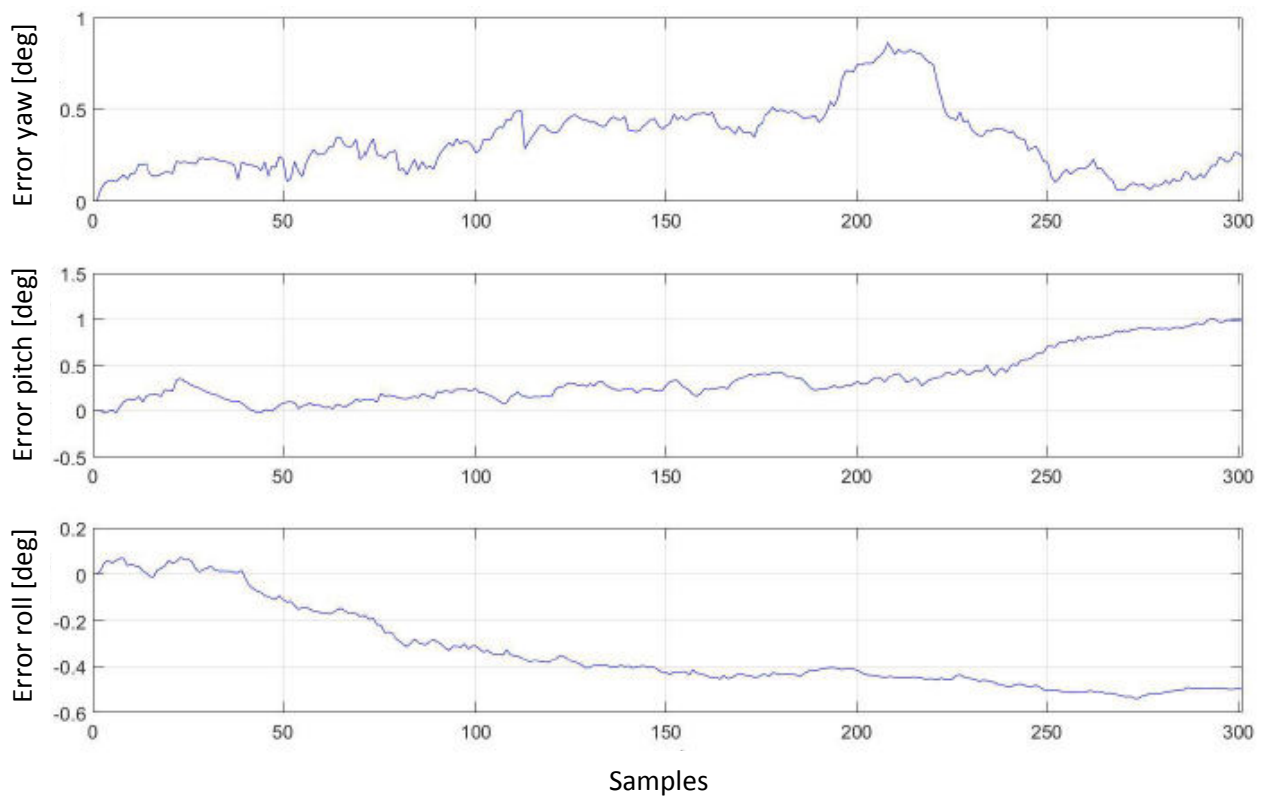


Figure 5.27: Cumulative errors for attitude in straight-line approach with stereo navigation filter.

The second test, with 2 Hz framerate has been analysed with a shorter travelled distance, i.e. from 12.5m to 5m. The estimated trajectory and attitude are represented in Figures 5.28 and 5.29.

Despite the shorter distance, the errors appear larger than for the previous case. This is related to the smaller inter-frame motion, which in this case is 2.5 cm.

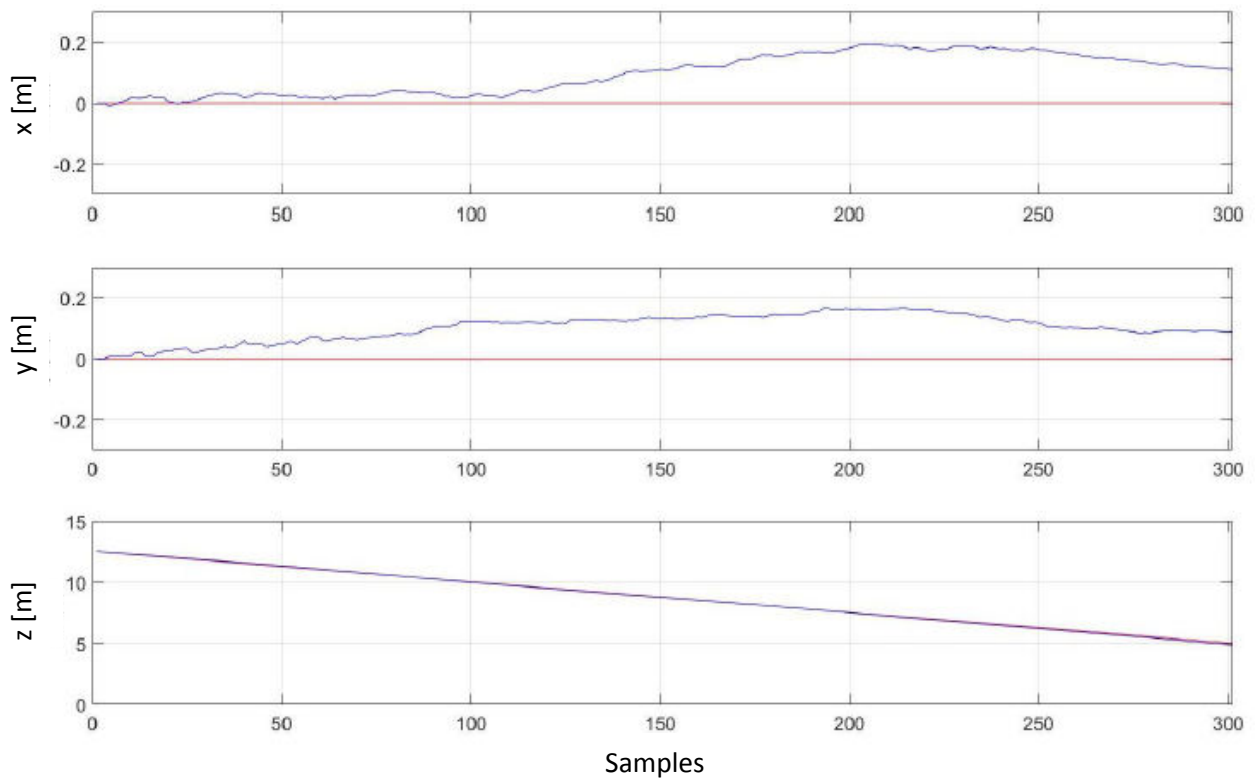


Figure 5.28: Estimated trajectory (blue) and ground truth (red) for straight-line approach with stereo navigation filter. Second test.

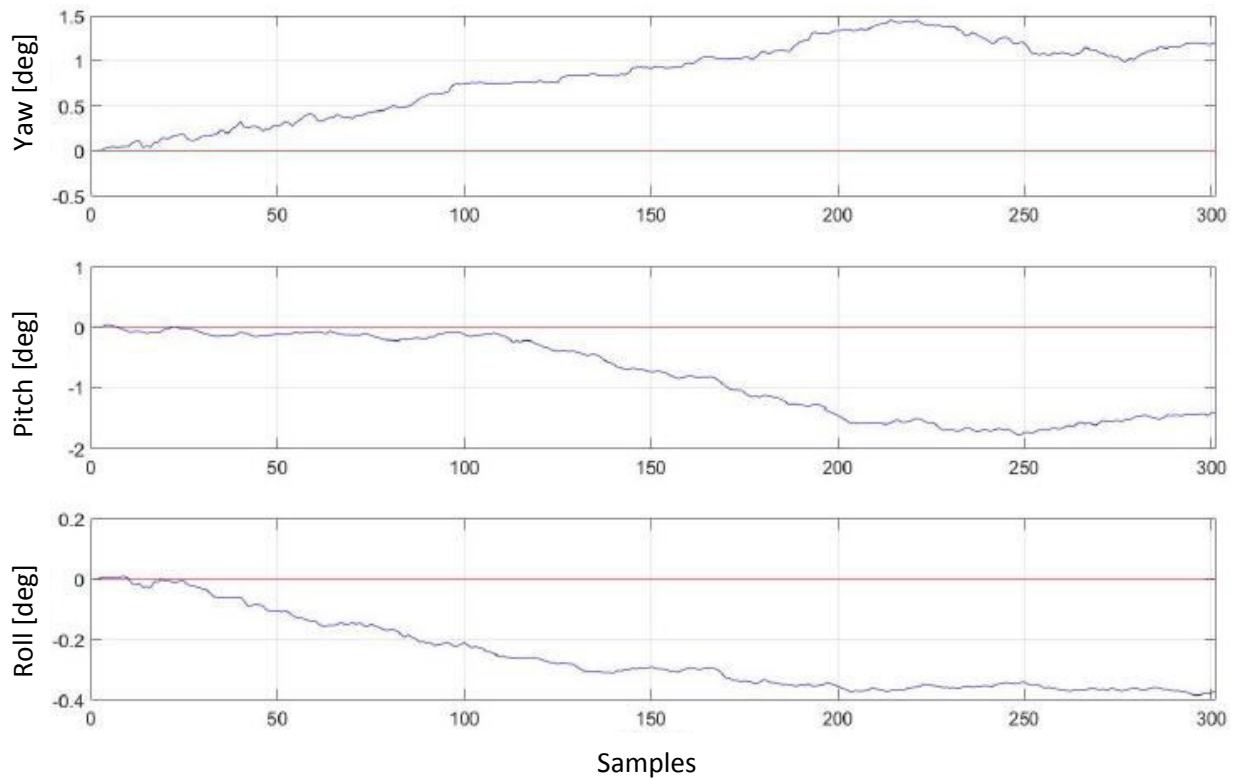


Figure 5.29: Estimated attitude (blue) and ground truth (red) for straight-line approach with stereo navigation filter. Second test.

The inter-frame motion errors are then analysed. Figure 5.30 shows the inter-frame errors for the trajectory. The errors fluctuate around zero and are small, but since the inter-frame motion is smaller as well, their significance has increased. Figure 5.31 shows the inter-frame errors for the attitude.

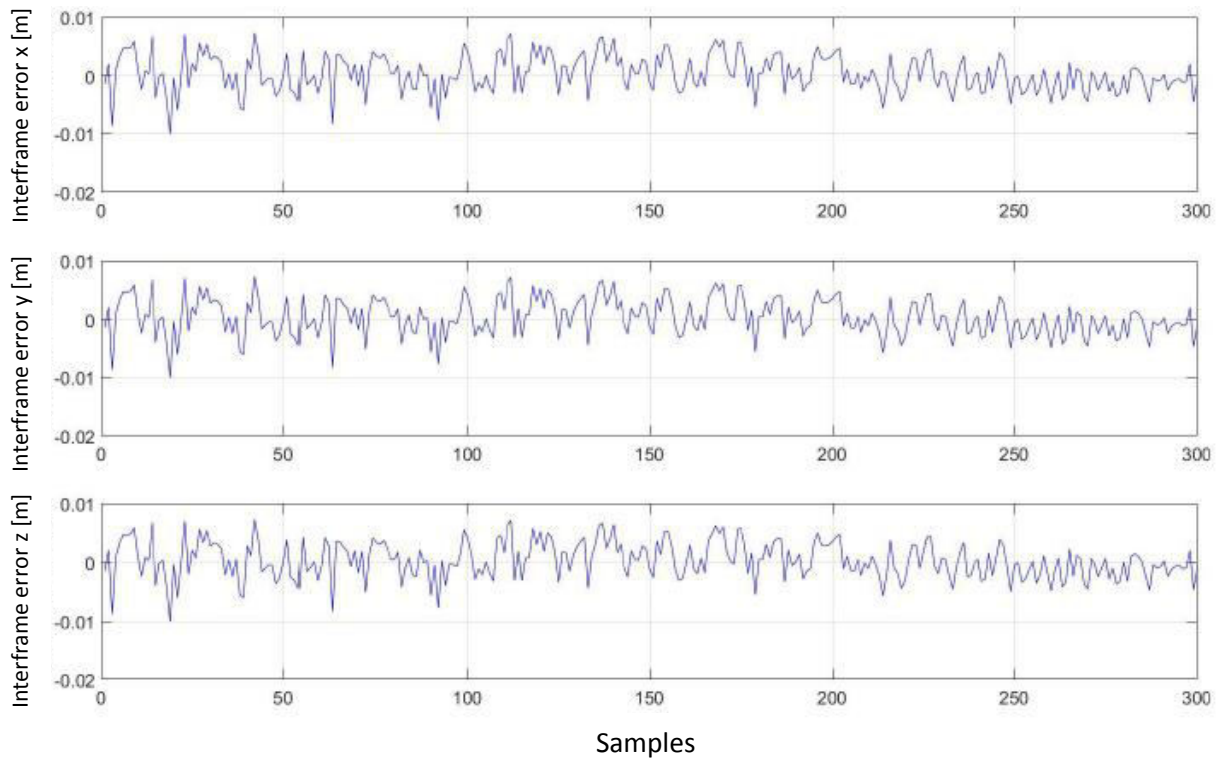


Figure 5.30: Inter-frame errors for trajectory in straight-line approach with stereo navigation filter. Second test.

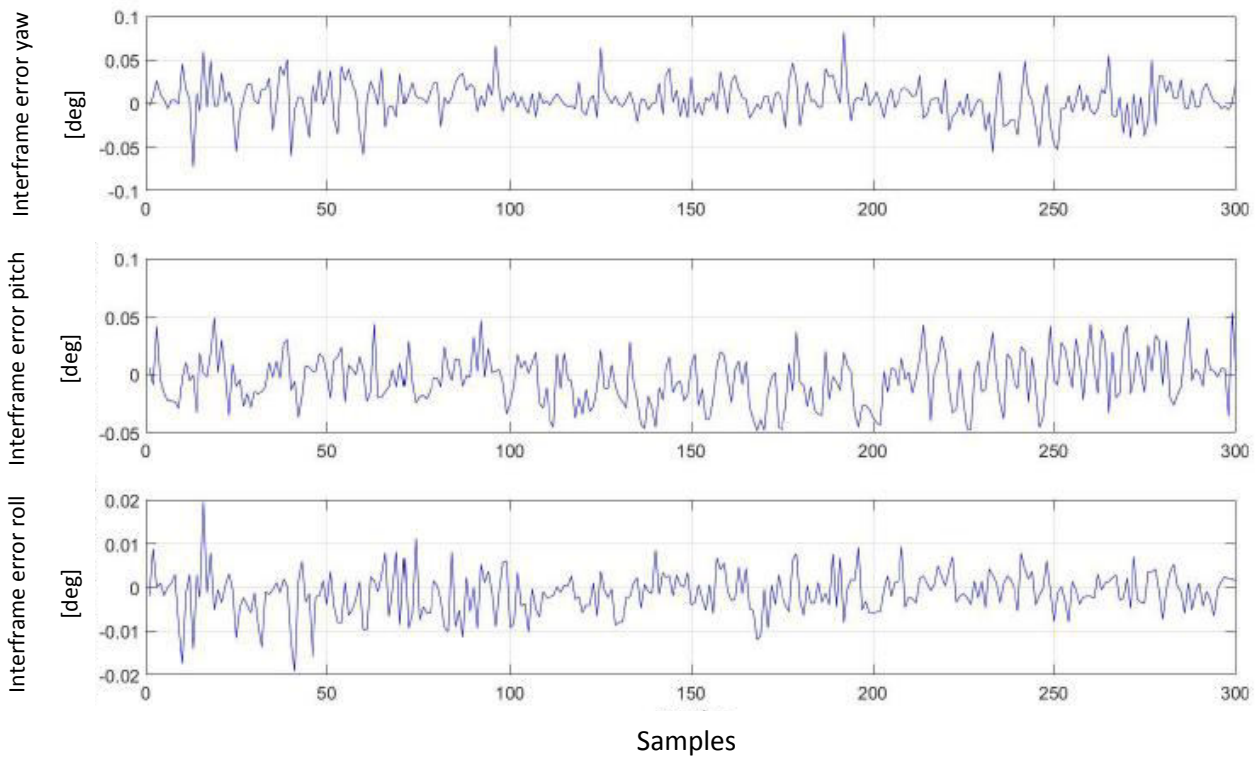


Figure 5.31: Inter-frame errors for attitude in straight-line approach with stereo navigation filter. Second test.

Figures 5.32 and 5.33 show the final errors for the trajectory and attitude, respectively. Despite the errors being small, since the inter-frame motion is smaller it was expected to result in a larger error with respect to the previous case due to the reduced capability in discrimination of very close motion. These results should serve as a way to improve the tuning.

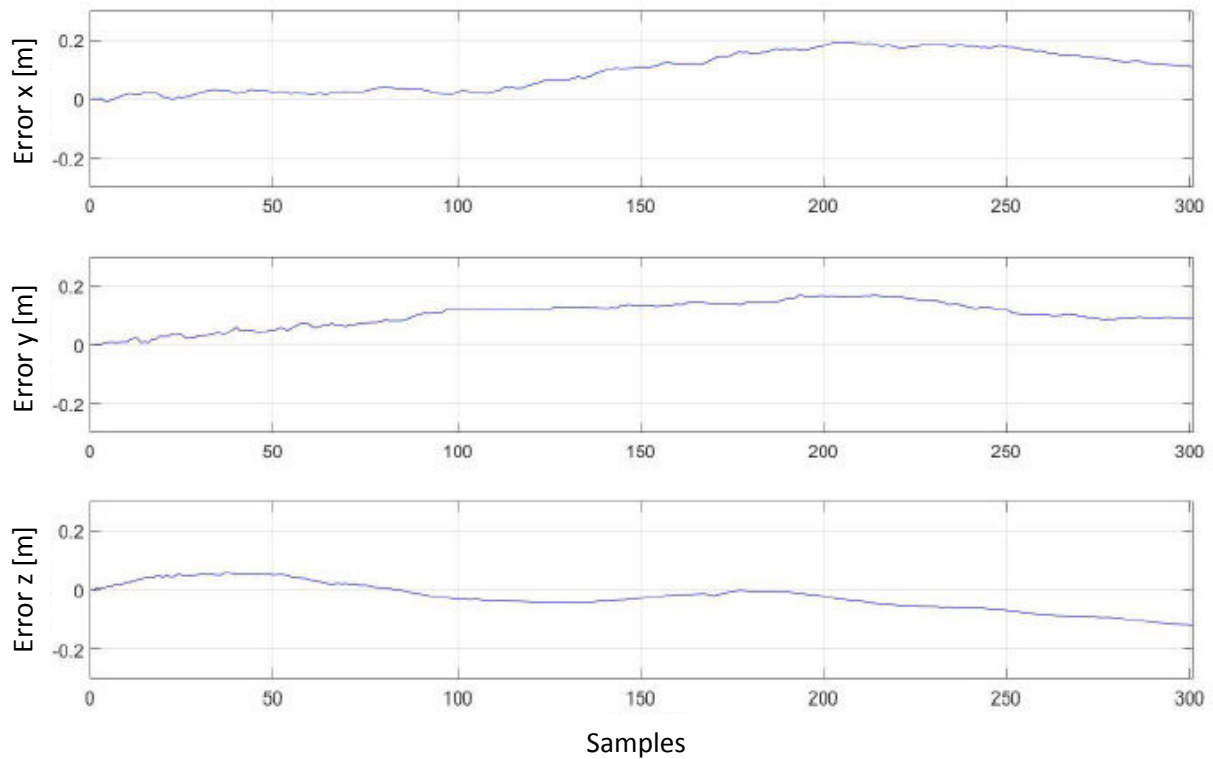


Figure 5.32: Cumulative errors for trajectory in straight-line approach with stereo navigation filter. Second test.

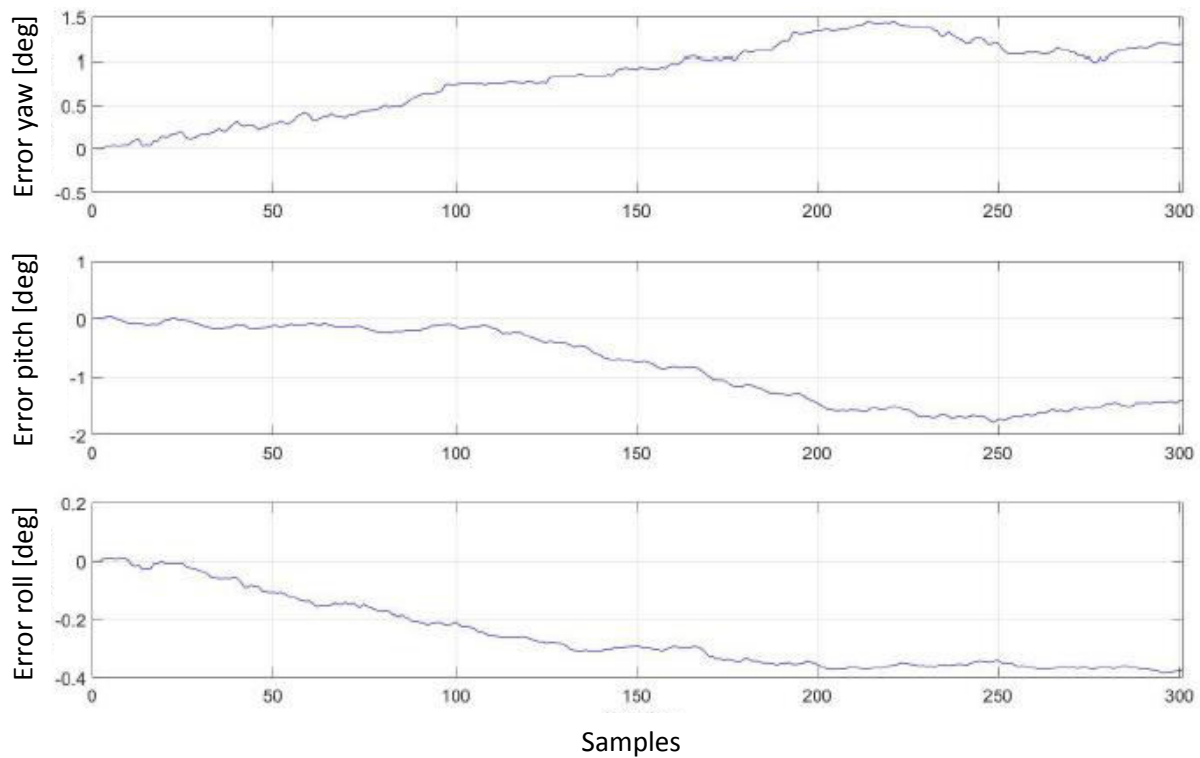


Figure 5.33: Cumulative errors for attitude in straight-line approach with stereo navigation filter. Second test.

The second analysed case is again a straight-line trajectory at a slower framerate and with the chaser keeping the distance and attitude stable in the last part. The mission profile is shown in Figures 5.34. The framerate is about 0.13 Hz.

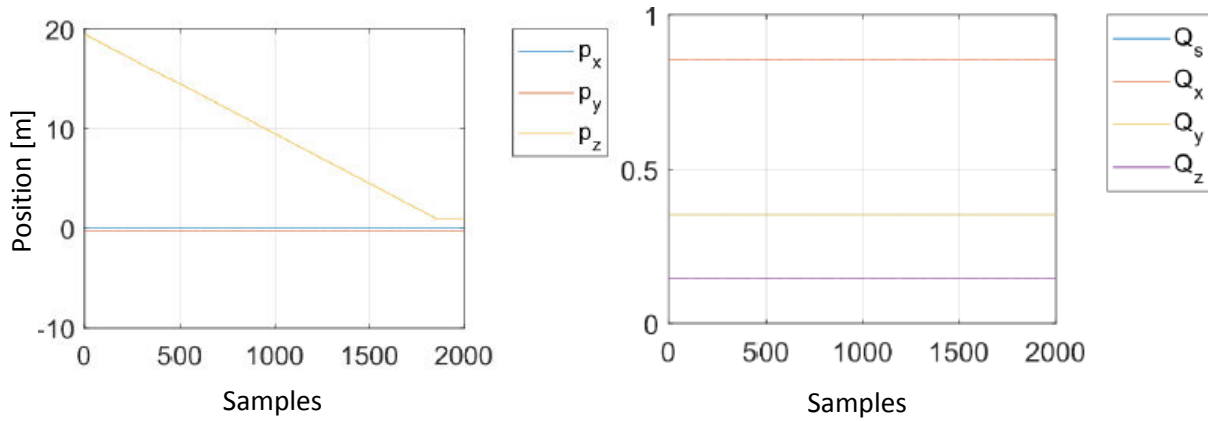


Figure 5.34: Translation and attitude profile mission.

The estimated trajectory, as shown in Figure 5.35, follows properly the ground truth tracks. A detailed analysis of the translation error as a function of covered distance (see Figure 5.36) shows a final error smaller than 1%.

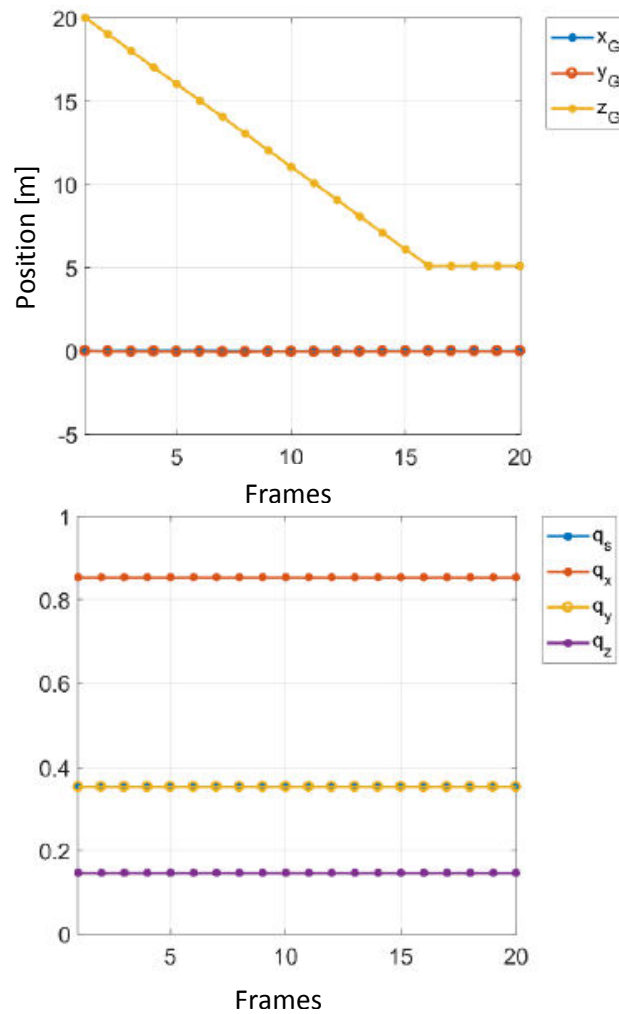


Figure 5.35: Reconstructed translation and rotation for the second navigation case.

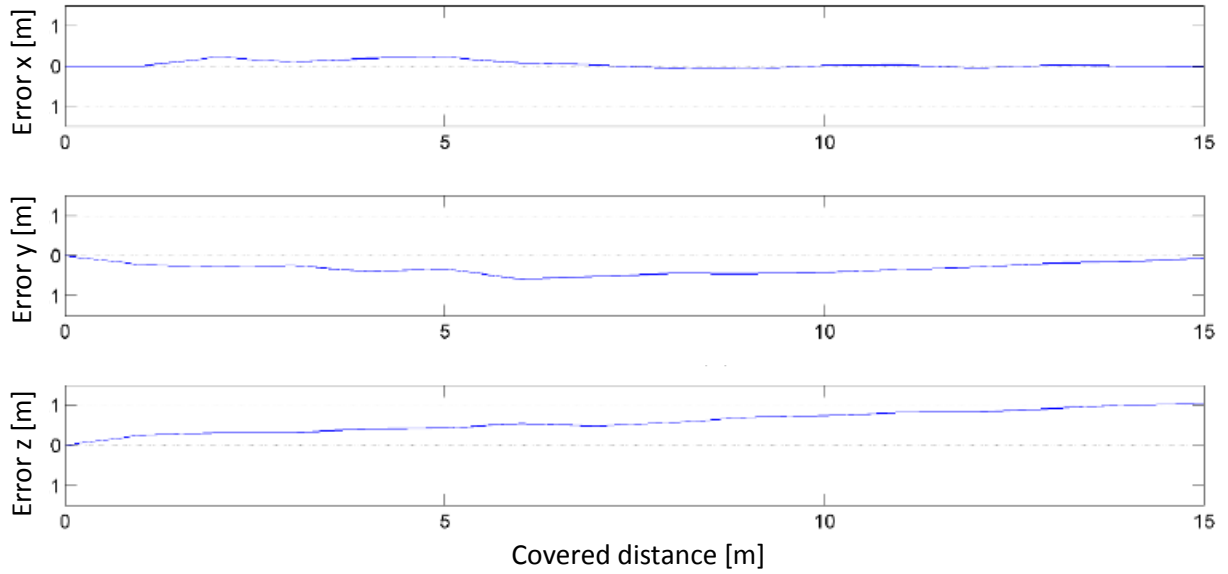


Figure 5.36: Translation error as function of covered distance for stereo navigation.

5.7. Overall computing times

Table 5.3 shows a summary of the computing times for the end-to-end data processing, from data acquisition to computation of the relative pose of the target with respect to the chaser satellite. The point cloud processing achieves the 0.5 fps requirement. However, the image processing does not meet the initial I3DS-defined 10 fps requirement.

Table 5.3: Computing times for all processing operations. The represented image processing computing times are for processing performed on the FPGA board. LiDAR processing has been done on the desktop computer.

Processing operation	Computing time per frame
Image pre-processing: Lens correction + Bilateral filtering + Histogram equalisation	0.16 s
Point cloud pre-processing: Uniform sampling + Statistical outlier removal	0.43 s
HR-camera-based navigation	1.32 s
Stereo-camera-based navigation	2.14 s
Point cloud-based navigation	0.57 s

Possible avenues to decrease the computing time for the image processing operations in order to meet the requirement, if it was not loosened because of the hardware limitations and if it was still not met on the actual I3DS FPGA board, are:

- Parallelisation on the FPGA board: The FPGAs structure allows for parallelisation, constituting one of the main advantages of FPGAs for their use in space. However, this capability has not been exploited during the development of this work and should be assessed in order to reduce the computing times. This could be done using only one FPGA board or incorporating more boards to the I3DS INSES.
- Transfer some of these operations to the OBC: the main reason to include an FPGA for data processing in the INSES is to distribute the computing load to achieve an overall faster multi-sensor processing chain that enables real-time operations. Therefore, the FPGA board

should represent an asset and not a limitation. Transferring some of the processing operations currently designed to be carried out on the FPGA to the more powerful OBC could thus ensure that the requirements are met.

- Further optimisation of the algorithms: another possible means to reduce the image processing computing time would be to revise the algorithms. They are currently based on the OpenCV library and the specific functions implemented by this library may not be the most efficient alternatives in terms of computing time.
- Last resort: in the event that none of these avenues was successful, decreasing the frequency of acquisition so that there is sufficient inter-frame time to achieve the end-to-end processing and navigation solution should be considered.

5.8. Conclusions

This Chapter has presented the tests performed to evaluate the end-to-end multi-sensor navigation solution, as well as the obtained results.

On-board the I3DS INSES, LiDAR data processing will be done on the OBC and image processing, on an FPGA. As such, both hardware platforms have been simulated in this work. The OBC has been simulated using a standard desktop computer and the FPGA has been simulated with the Xilinx UltraZed-EG FPGA board. The development environment has included the libraries OpenCV for image processing and PCL for point cloud processing. The FPGA development has been done using the PetaLinux Tools software package and the Xilinx SDK. All the necessary setup for FPGA development together with the required procedure have also been explained.

The second part of the Chapter represented the results of the study, including trajectory determination accuracy and computing times for both data pre-processing and navigation. A comparison between the processing performance of the desktop computer and the FPGA board in performing image pre-processing has also been included.

Finally, the overall computing times showed that the I3DS-defined time requirement for image processing on the FPGA board is not met at this stage of the research. Therefore, possible avenues to accelerate the processing are discussed. This could include parallelisation, further hardware and software optimisation, transferring some of the processing load to the OBC and decreasing the cameras acquisition frequency as a last resort.

Chapter 6 – Conclusions and future work

This work has been developed within the Integrated 3D Sensors (I3DS) project. I3DS is a project Co-funded under Horizon 2020 EU research and development program and part of the Strategic Research Cluster on Space Robotics Technologies. Its main ambition is to produce a standardised modular Inspector Sensor Suite (INSES) for autonomous orbital and planetary applications for future space missions.

A focus has been made on orbital applications, that can include scenarios such as on-orbit servicing and repair, space rendezvous and docking and active debris removal. The orbital sensor suite comprises a star tracker, a stereo camera, a high-resolution camera, a LiDAR and a high-frequency radar for relative navigation; a wide-angle illumination torch and a pattern projection sensor for illumination and a force-torque sensor and tactile sensor as contact sensors. All these sensors are interfaced through an Interface Control Unit (ICU).

In particular, this MSc by Research has tackled the problem of relative navigation for space applications using visual cameras, namely, a high-resolution camera and a stereo camera, and a LiDAR sensor. In this context, relative space navigation refers to the problem of estimating the relative pose (i.e. position and orientation) of a target spacecraft with respect to a chaser spacecraft on which the sensors will be mounted.

The system will be formed by an FPGA board to perform end-to-end image processing and an OBC for end-to-end LiDAR data processing. The motivation to include an FPGA in the design is to distribute the processing load, relieving some of it from the OBC to accelerate the whole processing pipeline with the intention to produce a solution for real-time applications. The main advantages of FPGAs for use in space are the parallelisation allowed by their architecture, which enables faster processing, and that they are reprogrammable, allowing software updates after launch. On the other hand, traditional FPGAs are susceptible to single-event upsets (SEUs) due to space radiation. However, companies such as Xilinx have developed space-rated FPGAs with a higher robustness to SEUs.

First, the data acquired by the sensors will be pre-processed in order to convert them into an optimum format for the posterior navigation algorithms. Image and point cloud distortion sources, pre-processing methods for both spatial and spectral correction and data down-sampling for the point clouds have been presented. Image pre-processing includes lens distortion correction, bilateral filtering and histogram equalisation for each monocular camera, with the stereo camera understood here as a set of two monocular cameras. Additionally, further pre-processing operations are performed on each stereo image pair in order to retrieve depth information and to create point clouds that later can potentially be integrated with data acquired by other sensors such as LiDAR or pattern projection for a fusion-based navigation. Point cloud pre-processing includes data down-sampling with a uniform sampling filter and statistical outlier removal.

Once the data have been pre-processed, they can be input to the navigation algorithms. Stereo-based navigation is performed by temporally matching successive pairs of previously rectified and matched stereo images. This information is then passed on to visual odometry to retrieve the pose of the target satellite. High-resolution navigation is done using the PnP algorithm, supported by two sets of known fiducial markers on the surface of the target satellite. Finally, four different algorithms

have been evaluated for LiDAR based navigation, namely, the classical Iterative Closest Point (ICP) point cloud registration algorithm, its variants Generalised ICP and Trimmed ICP and a Histogram of Distances (HoD) point descriptor, developed at Cranfield University, with a Kalman filter-based registration.

The FPGA board to be used within the INSES has been simulated by the Xilinx UltraZed-EG FPGA board and the OBC, by an 8 x Intel® Core™ i7-6700 CPU @ 3.40 GHz desktop computer with Ubuntu 16.04.4 LTS. As for the development environment, the OpenCV library has been used for image processing and the Point Cloud Library (PCL) library for point cloud processing. The image processing applications for the FPGA board have been developed on Xilinx Software Development Kit (SDK). The operating system on-board the FPGA board is Petalinux.

The algorithms have been tested on data simulated by Thales Alenia Space France (TAS-F). Images simulation has been done using the TAS-F-internal software SPICAM and LiDAR data have been simulated with the open-source software BlenSor. The data simulate a straight-line approach trajectory of the chaser towards a target satellite, from 20 m, without any rotational motion. The approach velocity is 0.05 m/s. This trajectory simulates the final approximation approach of a space rendezvous. The size of the raw LiDAR point clouds is 152100 points (5 MB) and that of the raw images is 2048 pixels x 2048 pixels (0.5 MB). Furthermore, the target satellite features fiducial markers on its surface to support monocular visual navigation.

For this trajectory, the best performance for point cloud-based navigation was achieved with the Generalised ICP algorithm. In particular, the trajectory determination error below 5 % for the three sensors. The end-to-end computing times (i.e. data pre-processing and navigation) were 1.48 seconds for HR-based navigation, 2.30 seconds for stereo-based navigation and 1 second for LiDAR-based navigation.

The initial I3DS-defined requirement for image processing was 10 fps, which has not been met. Possible avenues to meet this requirement could include taking profit of the parallelisation opportunities offered by FPGA technology, either on one single or on multiple FPGA boards, further optimisation of the algorithms and decreasing the acquisition frequency as last resource. It must also be considered that the Xilinx UltraZed-EG FPGA board used to simulate the on-board FPGA board has lower specifications than the board that will be installed on the final INSES. Therefore, further analysis with the appropriate hardware would be necessary to assess its real capabilities.

On the other hand, the acquisition rate of the LiDAR simulated by TAS-F is 0.5 fps, which would result in a 2-second computing time requirement for LiDAR processing, which is achieved.

A number of difficulties were encountered during the development of this project - the most outstanding one being the lack of experience in FPGA development within the Autonomous Systems group. After a steep learning curve, it is expected that this document can serve as an aid for future students. Figures 5.3 and 5.4 on the system setup as well as Section 5.4 on the test bench setup design, are particularly relevant and should be consulted for this purpose.

Overall, the main contribution of the research has been the development of an end-to-end navigation solution for space using an FPGA board, including data pre-processing and navigation algorithms. Contrary to the mission-specific Application Specific Integrated Circuits (ASICs) traditionally used in space missions, FPGAs would reduce mission costs as well as allow for in-flight

reprogrammability. This work demonstrates the possibility of substituting and/or complementing SoCs for relative navigation for space applications.

References

- [1] X. Clerc and I. Retat. "Astrium vision on space debris removal," in *Proc. 63rd International Astronautical Congress*, 2012, pp. 2873-2885.
- [2] C. Bonnal, J. M. Ruault and M. C. Desjean. "Active debris removal: recent progress and current trends." *Acta Astronautica*, vol. 85 (2013), pp. 51–60, Apr. 2013.
- [3] A. Tatsch, N. Fitz-Coy and S. Gladun. "On-orbit servicing: a brief survey." *Performance Metrics for Intelligent Systems Workshop*, Aug. 2006, Gaithersburg, MD, USA.
- [4] A. Flores-Abad, O. Ma, K. Pham and S. Ulrich. "A review of space robotics technologies for on-orbit servicing." *Progress in Aerospace Sciences*, vol. 68, pp. 1-26, Jul. 2014.
- [5] Polites M E. (1998, July). "An Assessment of the Technology of Automated Rendezvous and Capture in Space." Available: <https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19980219470.pdf> [Sep. 1, 2018]
- [6] D. Hoffpauir. "Tomorrow's technology will build on historical breakthroughs." Internet: <https://www.nasa.gov/offices/nesc/articles/space-rendezvous>, 23 May 2016 [Sep. 1, 2018].
- [7] European Space Agency. "Space debris by the numbers." Internet: https://www.esa.int/Our_Activities/Operations/Space_Debris/Space_debris_by_the_numbers, Jan. 2018 [Oct. 30, 2018].
- [8] L. M. B. Winternitz, W. A. Bamford and G. W. Heckler. "A GPS Receiver for High-Altitude Satellite Navigation." *IEEE Journal of Selected Topics in Signal Processing*, vol. 3(4), pp. 541-556, Aug. 2009.
- [9] I3DS Integrated 3D Sensors. "What's I3DS?" Internet: <http://i3ds-h2020.eu/about-us>, 2018 [20 Aug. 2018].
- [10] I3DS Integrated 3D Sensors. "Partners." Internet: <http://i3ds-h2020.eu/about-us/partners>, 2018 [Aug. 20, 2018].
- [11] S. Andiappane. (2017, Oct). "D2.4: Design Description Document, ICD Details." Available: http://i3ds-h2020.eu/images/OG4_I3DS_D2.4-Design_Description_Document_ICD-details_PublicRelease.pdf [Aug. 20, 2018].
- [12] Beamagine. "I3CAM Specifications." Internet: <http://beamagine.com/products/i3cam/>, 2017 [Aug. 20, 2018].
- [13] CMOSIS. "CMV400." Internet: http://www.cmosis.com/products/product_detail/cmv4000, 2018 [Aug. 20, 2018].
- [14] V. Dubanchet, P. Pellegrino and A. Biggio. (2017, Oct). "D1.2: Use-case Identification." Available: http://i3ds-h2020.eu/images/OG4_I3DS_D1.2-Use_case_Identification_PublicRelease.pdf [Aug. 21, 2018].
- [15] J. A. Christian and S. Cryan. "A Survey of LIDAR Technology and its Use in Spacecraft Relative Navigation." *AIAA Guidance, Navigation, and Control (GNC) Conference*, 19-22 Aug. 2013, Boston, MA, USA.

- [16] ESRI, Environmental Systems Research Institute. "What is lidar data?" Internet: <http://desktop.arcgis.com/en/arcmap/10.3/manage-data/las-dataset/what-is-lidar-data-.htm>, 2016 [Oct. 15 2018].
- [17] Velodyne LiDAR. "Datasheet. PUCK™ VLP-16. Real-time 3D LiDAR sensor." Available: https://velodynelidar.com/docs/datasheet/63-9229_Rev-H_Puck%20Datasheet_Web.pdf, 2018 [Oct. 12 2018]
- [18] Velodyne LiDAR. "User Data Examples." Internet: <https://velodynelidar.com/vlp-16.html>, 2018 [Oct. 12 2018]
- [19] D. M. Winker, "The CALIPSO Mission and Initial Observations of Aerosols and Clouds from CALIOP," in *Fourier Transform Spectroscopy/ Hyperspectral Imaging and Sounding of the Environment, OSA Technical Digest Series (CD)* (Optical Society of America, 2007), paper HTuD1.
- [20] CSR/TSGC. "ICESat / GLAS." Internet: <https://www.csr.utexas.edu/glas/>, Jan. 2003 [Oct. 30, 2018].
- [21] R. Rajeeesh and G. S. Dwarakish. "Satellite oceanography- A review," in Proc. International Conference on Water Resources, Coastal and Ocean Engineering, 2015, vol. 5, pp. 165-172.
- [22] QImaging. "Rolling Shutter vs. Global Shutter." Available: <https://www.qimaging.com/ccdorscmos/pdfs/RollingvsGlobalShutter.pdf>, 2014 [Oct. 13, 2018]
- [23] Bedros S. "Robotics – ME 5286". Class Lecture, topic: "Image Formation and Cameras." Computer Vision. College of Science&Engineering, University of Minnesota, 2017.
- [24] Sanchez-Gestido M, Janschek K, Schnitzer F, Sonnenburg A, Dyblenko S, Ahrns I, Haskamp C, Ludwig S, Schmidt U, Hartmann R, Gil-Fernandez J, Huertas I, Dubois-Matra O and Ortega G. "Image processing for relative navigation in space rendezvous, small body navigation and descend and landing for planets and asteroids." *1st Annual RPI Workshop on Image-Based Modeling and Navigation for Space Applications*. 4-5 Jun. 2018, Troy, NY.
- [25] H. Hinkel, S. Cryan, C. D'Souza and M. Strube. "NASA's Automated Rendezvous and Docking Sensor Development and Its Applicability to the GER." *Space Exploration International Conference*, 29-31 Oct. 2014, Strasbourg, France.
- [26] D. C. Woffinden and D. K. Geller. "Navigating the Road to Autonomous Orbital Rendezvous." *Journal of Spacecraft and Rockets*, vol. 44(4), pp. 898-909, Jul.–Aug. 2007.
- [27] R. Rajesh. "Computer Vision – CSE455". Class Lecture, topic: "Stereo and 3D Vision." UW CSE vision faculty. Paul G. Allen School of Computer Science and Engineering, Seattle, WA, USA.
- [28] D. Fourie, B. E. Tweddle, S. Ulrich and A. Saenz-Otero. "Vision-Based Relative Navigation and Control for Autonomous Spacecraft Inspection of an Unknown Object." *AIAA Guidance, Navigation, and Control (GNC) Conference*, 19-22 Aug. 2013, Boston, MA, USA.
- [29] A. Fernández León. (2013, Apr.). "Trends and patterns of ASIC and FPGA use in European space missions." Available at: https://amstel.estec.esa.int/tecedm/website/stag_ygt/Boada.pdf [Oct.15 2018].

- [30] B. Joan. "Difference Between ASIC and FPGA." Internet: <http://www.differencebetween.net/technology/difference-between-asic-and-fpga/>, 9 Dec. 2009 [Oct. 15, 2018].
- [31] ESA, European Space Agency. "The use of reprogrammable FPGAs in space." Internet: https://www.esa.int/Our_Activities/Space_Engineering_Technology/Microelectronics/The_use_of_reprogrammable_FPGAs_in_space, 29 Apr. 2014 [Oct. 15, 2018].
- [32] S. Habinc. (2002, Sep.). "Suitability of reprogrammable FPGAs in space applications." Available: http://microelectronics.esa.int/techno/fpga_002_01-0-4.pdf [Accessed: October 2018]
- [33] P. Pingree. "Advancing NASA's on-board processing capabilities with reconfigurable FPGA technologies: Opportunities & implications." *IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*, 19-23 Apr. 2010, Atlanta, GA, USA.
- [34] E. Hecht. *Optics*. Essex, UK: Pearson Education Limited, 2017. 5th Edition.
- [35] S. Bedros. ME 5286: Robotics. Class Lecture, Topic: "Image Formation and Cameras." College of Science&Engineering, University of Minnesota. 2017.
- [36] R. Szeliski. *Computer Vision: Algorithms and Applications*. London, UK: Springer Verlag London, 2011.
- [37] L. M. Feetham, "Robust Vision-Based Slope Estimation and Rocks Detection for Autonomous Space Landers," PhD, Cranfield University, 2017.
- [38] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998.
- [39] OpenCV Dev. Team. "Camera Calibration and 3D Reconstruction." Internet: https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html, 2011-2014 [Oct. 19, 2018].
- [40] D. B. Goldman and J.H. Chen. "Vignette and Exposure Calibration and Compensation." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32(12), pp. 2276–2288, Dec. 2010.
- [41] J. J. G. Savard. "The Five Seidel Aberrations." Internet: <http://www.quadibloc.com/science/opt0505.htm>, 2001-2004 [Oct. 17, 2018].
- [42] Cambridge in Colour. "Camera lens corrections." Internet: <https://www.cambridgeincolour.com/tutorials/lens-corrections.htm>, 2005-2008 [Oct. 17, 2018].
- [43] Y. Zheng, S. Lin and S. B. Kang. "Single-Image Vignetting Correction." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31(12), pp. 2243 – 2256, Dec. 2009.
- [44] A. Kordecki, H. Palus and A. Bal. "Practical vignetting correction method for digital camera with measurement of surface luminance distribution." *Signal, Image and Video Processing*, vol. 10(8), pp. 1417–1424, Nov. 2016.

- [45] Y. Jin, L. M. Fayad and A. F. Laine. "Contrast enhancement by multiscale adaptive histogram equalization," in *Proc. SPIE 4478, Wavelets: Applications in Signal and Image Processing IX*, 2001, pp. 206-213.
- [46] B. Tarika, M. Kaur and I. Singh. "Review of Histogram Equalization Techniques." *International Journal Of Engineering And Computer Science*, vol. 3 (11), pp. 9205-9210, Nov. 2014.
- [47] S. M. Pizer and E. P. Amburn. "Adaptive histogram equalization and its variations." *Computer Vision, Graphics, and Image Processing*, vol. 39, pp. 355-368, 1987.
- [48] Tomasi, C., and R. Manduchi. "Bilateral Filtering for Gray and Color Images." *ICCV'98 Proceedings of the Sixth International Conference on Computer Vision*, 1998.
- [49] B. Fisher. "Epipolar Geometry." Internet:
http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/EPSRC_SSAZ/node18.html, 1997 [Oct. 18, 2018].
- [50] C. Loop and Z. Zhang. "Computing Rectifying Homographies for Stereo Vision," in *Proc. 1999 IEEE Conf. Computer Vision and Pattern Recognition*, 1999, pp. 125-131.
- [51] H. Su and B. He. "Stereo rectification of calibrated image pairs based on geometric transformation." *I. J. Modern Education and Computer Science*, vol. 4, pp. 17-24, Jul. 2011.
- [52] N. Ayache and C. Hansen. "Rectification of image for binocular and trinocular stereovision," in *Proc. of 9th International Conference on Pattern Recognition*, 1988, pp. 11-16.
- [53] A. Fusiello, E. Trucco and A. Verri. "A compact algorithm for rectification of stereo pairs." *Machine Vision and Applications*, vol. 12, pp.16–22, Jul. 2000.
- [54] R. I. Hartley. "Theory and Practice of Projective Rectification." *International Journal of Computer Vision*, vol. 35(2), pp 115–127, Nov. 1999.
- [55] D. G. Lowe. "Distinctive image features from scale-invariant keypoints." *International Journal of Computer Vision*, vol. 60(2), pp. 91-110, Nov. 2004.
- [56] H. Bay, T. Tuytelaars, and L. Van Gool. "Surf: Speeded up robust features," in *Proc. European Conference on Computer Vision*, 2006, pp. 404-417.
- [57] E. Rublee, V. Rabaud, K. Konolige and G. Bradski. "ORB: an efficient alternative to SIFT or SURF," in *Proc. International Conference on Computer Vision*, 2011.
- [58] E. Rosten and T. Drummond. "Machine learning for highspeed corner detection," in *Proc. European Conference on Computer Vision*, 2006, pp. 430-443.
- [59] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. "BRIEF: Binary robust independent elementary features," in *Proc. European Conference on Computer Vision*, 2010, pp. 778-792.
- [60] E. Karami, S. Prasad and M. Shehata. "Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images." *2015 Newfoundland Electrical and Computer Engineering Conference*, 5-6 Nov. 2015, St. John's, Canada.
- [61] M. Muja. "FLANN - Fast Library for Approximate Nearest Neighbors." Internet:
<https://www.cs.ubc.ca/research/flann/>, n.d. [Oct. 19, 2018].

- [62] M. Muja and D. G. Lowe. "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration." *International Conference on Computer Vision Theory and Applications*, 5-8 Feb. 2009, Lisboa, Portugal.
- [63] B. D. Lucas and T. Kanade. "An iterative image registration technique with an application to stereo vision," in *Proc. Imaging Understanding Workshop*, 1981, pp. 121-130.
- [64] C. Tomasi and T. Kanade, "Detection and tracking of point features", *Carnegie Mellon University Technical Report CMU-CS-91-132*, April 1991.
- [65] J. Shi and C. Tomasi, "Good features to track," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1994, pp. 593-600.
- [66] R. I. Hartley and P. Sturm. "Triangulation." *Computer Vision and Image Understanding*, vol. 68(2), pp. 146-157, Nov. 1997.
- [67] R. Hartley, R. Gupta, and T. Chang. "Stereo From Uncalibrated Cameras," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1992, pp. 761–764
- [68] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. New York, US: Cambridge University Press, 2004.
- [69] H. Hirschmuller, "Accurate and efficient stereo processing by semi-global matching and mutual information," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2005, vol. 2, pp. 807–814.
- [70] P. Viola and W. M. Wells III. "Alignment by Maximization of Mutual Information." *International Journal of Computer Vision*, vol. 24(2), pp. 137–154, Sep. 1997.
- [71] D. Hawkins. *Identification of Outliers*. Netherlands: Springer Netherlands, 1980.
- [72] S. Sotoodeh. "Outlier detection in laser scanner point clouds." *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 36(5), pp. 297-302.
- [73] K. Wolff, C. Kim, H. Zimmer, C. Schroers, M. Botsch, O. Sorkine-Hornung and A. Sorkine-Hornung. "Point Cloud Noise and Outlier Removal for Image-Based 3D Reconstruction," in *Proc. International Conference on 3D Vision (3DV)*, 2016, pp. 118-127.
- [74] J. A. Beraldin. "Integration of laser scanning and close-range photogrammetry – The last decade and beyond," in *XXth International Society for Photogrammetry and Remote Sensing (ISPRS) Congress. Commission VII*, 2004, pp. 972-983.
- [75] B. Lohani. "LiDAR error sources" Internet:
http://home.iitk.ac.in/~blohani/LiDAR_Tutorial/LiDARerrorsources.htm, 2010 [Oct. 20, 2018].
- [76] S. Papadimitriou, H. Kitagawa, P. B. Gibbons and C. Faloutsos. "LOCI: fast outlier detection using the local correlation integral," in *Proc. 19th International Conference on Data Engineering*, 2003, pp. 315-326.
- [77] M. M. Breunig, H. P. Kriegel, R. T. Ng, J. Sander. "LOF: Identifying Density-Based Local Outliers," in *Proc. ACM SIGMOD 2000 Int. Conf. On Management of Data*, 2000, pp. 93-104.

- [78] PointClouds.org. "Removing outliers using a StatisticalOutlierRemoval filter" Internet: http://pointclouds.org/documentation/tutorials/statistical_outlier.php, n.d. [Oct. 20, 2018].
- [79] PointClouds.org. "Uniform Sampling" Internet: http://docs.pointclouds.org/1.7.1/classpcl_1_1_uniform_sampling.html, 2013. [Oct. 20, 2018].
- [80] S. Orts-Escolano, V. Morell, J. García-Rodríguez and M. Cazorla. "Point Cloud Data Filtering and Downsampling using Growing Neural Gas," in *2013 International Joint Conference on Neural Networks*, 4-9 Aug. 2013, Dallas, Texas, US.
- [81] H. Wu, R. Chellappa, A. C. Sankaranarayanan and S. K. Zhou. "Robust Visual Tracking Using the Time-Reversibility Constraint." *IEEE International Conference on Computer Vision*, 14-21 Oct. 2007, Rio de Janeiro, Brazil.
- [82] B. Kitt, A. Geiger and H. Lategahn. "Visual odometry based on stereo image sequences with RANSAC-based outlier rejection scheme," in *Proc. IEEE Intelligent Vehicles Symposium*, 2010, pp. 486-492.
- [83] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas and M. J. Marín-Jiménez. "Automatic generation and detection of highly reliable fiducial markers under occlusion." *Pattern Recognition*, vol. 47(6), pp. 2280-2292, Jun. 2014.
- [84] D. G. Lowe. "Object recognition from local scale-invariant features," in *Proc. Seventh IEEE International Conference on Computer Vision*, 1999, pp. 1150-1157.
- [85] A. Segal, D. Haehnel and S. Thrun. "Generalized-ICP." *Robotics: Science and Systems*, June 28 - July 1, 2009 Seattle, WA, USA.
- [86] P. Besl and N. McKay. "A Method for Registration of 3-D Shapes." *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 14(2) pp. 239-256, Feb. 1992.
- [87] D. Chetverikov, D. Svirko, D. Stepanov and P. Krsek. "The Trimmed Iterative Closest Point algorithm," in *Proc. Object recognition supported by user interaction for service robots*, 2002 (3), pp. 545-548.
- [88] S. Rusinkiewicz and M. Levoy. "Efficient variants of the ICP algorithm," in *Proc. International Conference on 3-D Digital Imaging and Modeling*, 2002, pp. 145-152.
- [89] Y. Chen and G. Medioni. "Object Modeling by Registration of Multiple Range Images," in *Proc. of the 1992 IEEE Intl. Conf. on Robotics and Automation*, 1991, pp. 2724-2729.
- [90] S. Granger, X. Pennec. "Multi-scale EM-ICP: A Fast and Robust Approach for Surface Registration," in *Proc. of 7th European Conference on Computer Vision*, vol. 4, 2002, pp. 69-73, 2002.
- [91] A. Johnson and S. Kang. "Registration and Integration of Textured 3-D Data," *Proc. International Conference on 3-D Digital Imaging and Modeling*, 1997, pp. 234-241.

- [92] B. Bellekens, V. Spruyt, R. Berkvens and M. Weyn. "A Survey of Rigid 3D Pointcloud Registration Algorithms," in *Proc. International Conference on Ambient Computing, Applications, Services and Technologies*, 2014, pp. 8-13.
- [93] O. Kechagias-Stamatis and N. Aouf. "Histogram of distances for local surface description," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 2487-2493.
- [101] Universität Salzburg. "BlenSor. Blender Sensor Simulator." Internet: <http://www.blensor.org/>, 2018 [Oct. 27, 2018].
- [102] Avnet. "UltraZed-EG™ SOM." Available: http://zedboard.org/sites/default/files/product_briefs/5043-PB-AES-ZU3EG-1-SOM-G-V3.pdf, 2017 [Oct. 27, 2018].
- [103] Xilinx. "Space." Internet: <https://www.xilinx.com/applications/aerospace-and-defense/space.html> , 2018 [Oct. 27, 2018].
- [104] Avnet. "UltraZed. UltraZed SOMs". Internet: <http://zedboard.org/product/UltraZed>, 2018 [Oct. 28, 2018].
- [105] OpenCV team. "About. OpenCV." Internet: <https://opencv.org/about.html>, 2018 [Oct. 27, 2018].
- [106] R. B. Rusu and S. Cousins. "3D is here: Point Cloud Library (PCL)." Available: http://pointclouds.org/assets/pdf/pcl_icra2011.pdf, 2011 [Oct. 27, 2018].
- [107] Xilinx. "Xilinx Software Development Kit (XSDK)." Internet: <https://www.xilinx.com/products/design-tools/embedded-software/sdk.html>, 2018 [Oct. 27, 2018].
- [108] Xilinx. "PetaLinux Tools Documentation. Reference Guide." Available: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_4/ug1144-PetaLinux-tools-reference-guide.pdf, 20th Dec. 2017 [Oct. 27, 2018].
- [109] Xilinx. "PetaLinux Tools." Internet: <https://www.xilinx.com/products/design-tools/embedded-software/PetaLinux-sdk.html>, 2018 [Oct. 27, 2018].
- [110] C. M. Penalver. "Minicom. Ubuntu documentation." Internet: <https://help.ubuntu.com/community/Minicom>, 20th Jan. 2015 [Oct. 28, 2018].
- [111] M. van Smoorenburg, M. K. Johnson and J. Lahtinen. "minicom(1) - Linux man page." Internet: <https://linux.die.net/man/1/minicom>, 2017 [Oct. 28, 2018].
- [112] PointClouds.org. "About. What is PCL?". Internet: <http://pointclouds.org/about/>, 2018 [Oct. 28, 2018].
- [113] PointClouds.org. "Documentation. Building PCL's dependencies from source on Windows". Internet: http://pointclouds.org/documentation/tutorials/compiling_pcl_dependencies_windows.php#compiling-pcl-dependencies-windows, 2018 [Oct. 28, 2018].

ANNEX – Experimental work

During the development of this MSc by Research some experimental work was done. This included LiDAR data acquisition in the Autonomous System Group's laboratory, processing of the LiDAR data and tuning of the LiDAR-based-navigation filters. However, due to a problem with the data acquisition these data never produced satisfactory results and therefore were not included in the main body of the present dissertation. Yet, the amount of time and effort invested in this process was considerably high and therefore it has been thought appropriate to include an annex documenting it.

Data acquisition

All the experimental work was performed using the LiDAR sensor Velodyne LiDAR Puck VLP-16. This LiDAR sensor features 16 channels with a 360° horizontal FoV and $\pm 15^\circ$ vertical FoV. It can acquire up to 600 000 points per second, has an 8 W typical consumption and a 5 Hz – 20 Hz rotation rate. It operates on a 903 nm wavelength. Its mass is 830 grams. More information and documentation about this sensor can be found on: <https://velodynelidar.com/vlp-16.html>.

The motion capture system used to obtain the ground truth for navigation error analysis was Optitrack (<http://optitrack.com/>). Optitrack is an optical motion capture system composed of various IR cameras taking 2-D images. The system calculates the 3-D coordinates of the tracked object via triangulation by comparing the overlapping areas in the 2-D images. For this purpose, a minimum of three IR markers must be placed on the object to track in order to define a rigid body recognisable by Optitrack. The data captured by the cameras are streamed to a working station where they are processed using the software program Motive. The system can locate the tracked object with a mm-accuracy.

The work environment for this research was the Autonomous Systems group's laboratory, which is a rectangular room with various objects. The test target satellite was an EnviSat model. To be able to establish a correspondence between the Optitrack measurements and the LiDAR reference system, the origin of the Optitrack reference system was situated below the EnviSat model. For the Optitrack system to be able to track the LiDAR movement, a rigid body was defined by placing three IR markers in a straight line just on top of the LiDAR, using a metal rectangular plate mounted on the LiDAR. Figure 1 shows a schematic diagram of this installation. During the tests, the LiDAR was placed on top of a wheeled trolley available in the laboratory and able to move in the space surrounding the EnviSat model. This posed the inconvenience that the movement was not fully stable, so the measurements were somewhat “bumpy”.

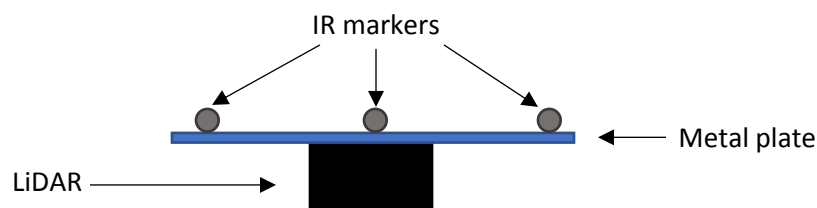


Figure 1: Markers installation on the LiDAR sensor.

In order to obtain correlated data that can be synchronised, both LiDAR and Optitrack systems had to start to record data at the same time. This was done using a ROS (Robotic Operating System). Each

sequence was recorded as a bag file from which the Optitrack odometry and the LiDAR point clouds could be retrieved. The point clouds recorded by the LiDAR were automatically named after the time when they were acquired. The Optitrack odometry contained the acquisition time, position and orientation of the rigid body defined during the Optitrack calibration, which in this case was the tree IR markers representing the LiDAR sensor. The orientation in this file was respective to the initial orientation (at the time of the Optitrack calibration). Thus, in order to be able to correlate the two reference systems from the initial time, the axis of both had to be aligned during the Optitrack calibration.

Besides both systems starting recording data at the same time, which usually is not exactly achieved, the recording frequency of the two systems is different. Thus, in order to be able to compare the pose estimation results obtained from the LiDAR data with the ground truth obtained with Optitrack, the data had to be synchronized. A C++ program was developed for this task. The program reads the name of the point clouds, which corresponds to the acquisition time, and uses it to look for the entry with the closest acquisition time in the Optitrack odometry file. It then saves the corresponding data (time, position and orientation) into a csv output file. This file will be referred to as the “synchronisation file”.

The different trajectories were:

- Trajectory 1: A straight line towards the target satellite up to a distance of 75 cm from it and then away from the satellite until reaching approximately the original position. This trajectory could represent the final approximation phase in a space rendezvous.
- Trajectory 2: A curved trajectory towards and away from the satellite, with a sharp turn in the closest point to the satellite. The aim of this test was to evaluate the ability of the navigation algorithm to keep tracking the target satellite when changing the viewing angle.
- Trajectory 3: This trajectory was designed to simulate a part of an orbit of the sensor around the target satellite. As such, it has a quasi-elliptical trace. This would represent the observation stage of a space rendezvous, before the final approach.
- Trajectory 4: A quasi-straight line perpendicular to the shortest line from the satellite to the plane of movement of the sensor.
- Trajectory 5: Same as trajectory 4 but at a closer distance to the target satellite. Because of the limited $\pm 15^\circ$ vertical FoV of the LiDAR sensor, the data acquired from this distance did not cover the whole satellite. The aim of this trajectory was to evaluate the ability of the algorithm to track the target satellite when not all of it was in the FoV of the sensor.

Figures 2 to 6 represent these trajectories as retrieved from the data acquired by Optitrack. The figures are expressed in Optitrack coordinate frame. In the figures, the blue line represents the movement of the LiDAR sensor and the red dot the position of the Envisat model. It must be noted that, as mentioned above, the origin of the Optitrack coordinate frame is placed just below the satellite and therefore the red dot has coordinates (0,0) in all figures.

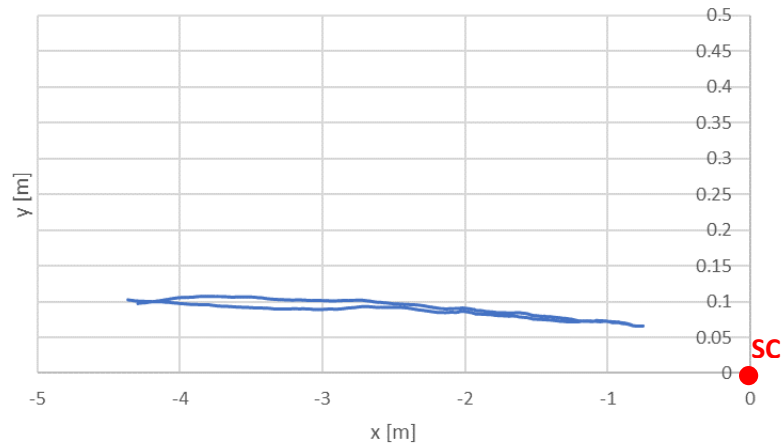


Figure 2: Trajectory 1 as retrieved from Optitrack data.

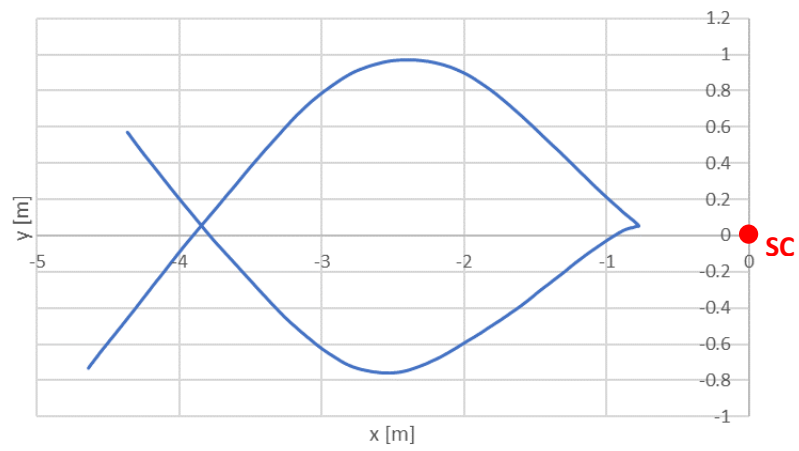


Figure 3: Trajectory 2 as retrieved from Optitrack data.

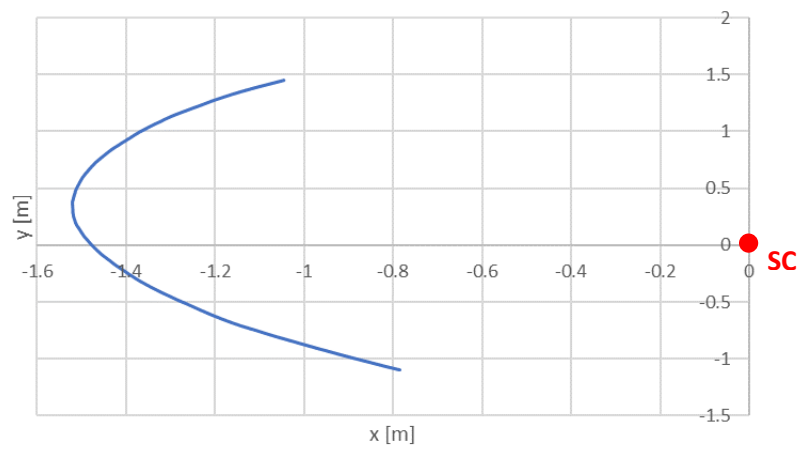


Figure 4: Trajectory 3 as retrieved from Optitrack data.

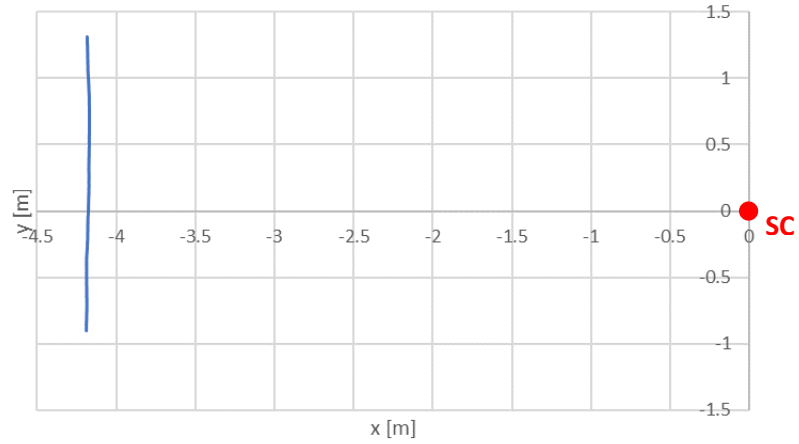


Figure 5: Trajectory 4 as retrieved from Optitrack data.

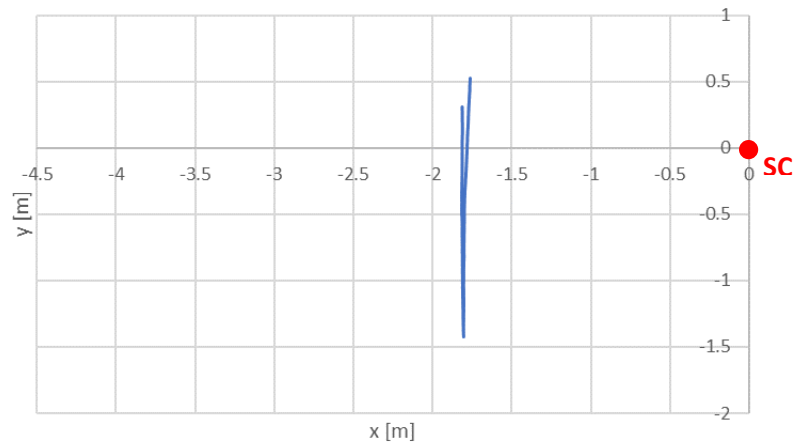
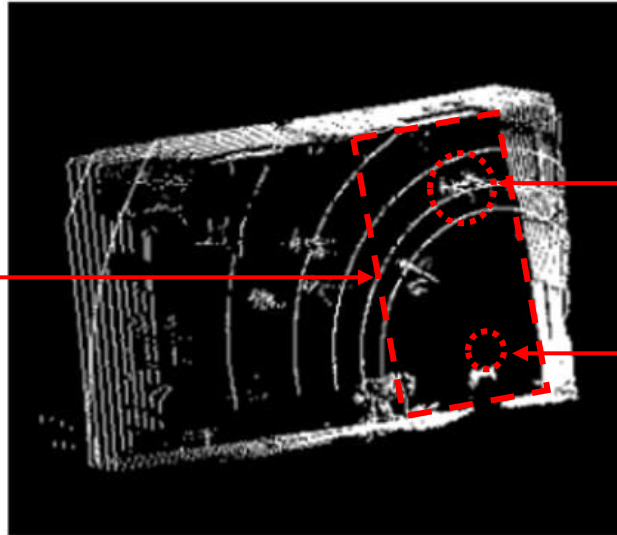


Figure 6: Trajectory 5 as retrieved from Optitrack data.

Figure 7 represents an example of a point cloud taken by the LiDAR sensor in the laboratory environment.

Area of coverage of the Optitrack motion capture system. The six cameras were situated approximately around the rectangle.



Envisat model

LiDAR

Figure 7: Point cloud acquired by the LiDAR sensor in the Autonomous Systems Group's laboratory.

Data processing

In order to better simulate the space environment where the only object reflecting the LiDAR laser beams will be the target, a point cloud pre-processing step was carried out in which the Envisat model was isolated from the rest of the laboratory environment.

This was achieved through the following steps:

1. The position and orientation of the LiDAR in the Optitrack reference system are retrieved from the file output of the *synchronisation file*. This is, the file with the Optitrack entries corresponding to those acquired at the same time than the LiDAR scans.
2. For each point cloud, the position vector of the LiDAR in the Optitrack reference system is rotated to the initial orientation using the quaternions contained in the *synchronisation file*.
3. The rotated position vector is inverted so as to represent the position of the Optitrack reference system origin in the LiDAR reference system.
4. Having the position of the origin of the Optitrack reference system (which is also the position of the satellite), the points pertaining to the satellite body in the point cloud can be isolated by discarding all the points outside of a bounding box around the satellite.

This process assumes that in the very first frame the axis of the LiDAR and Optitrack reference systems are aligned and their conversion is known.

The satellite isolation filter produced satisfactory results for all trajectories, which validated the algorithm. An example of applying this filter to the whole laboratory point cloud is depicted in Figure 8.

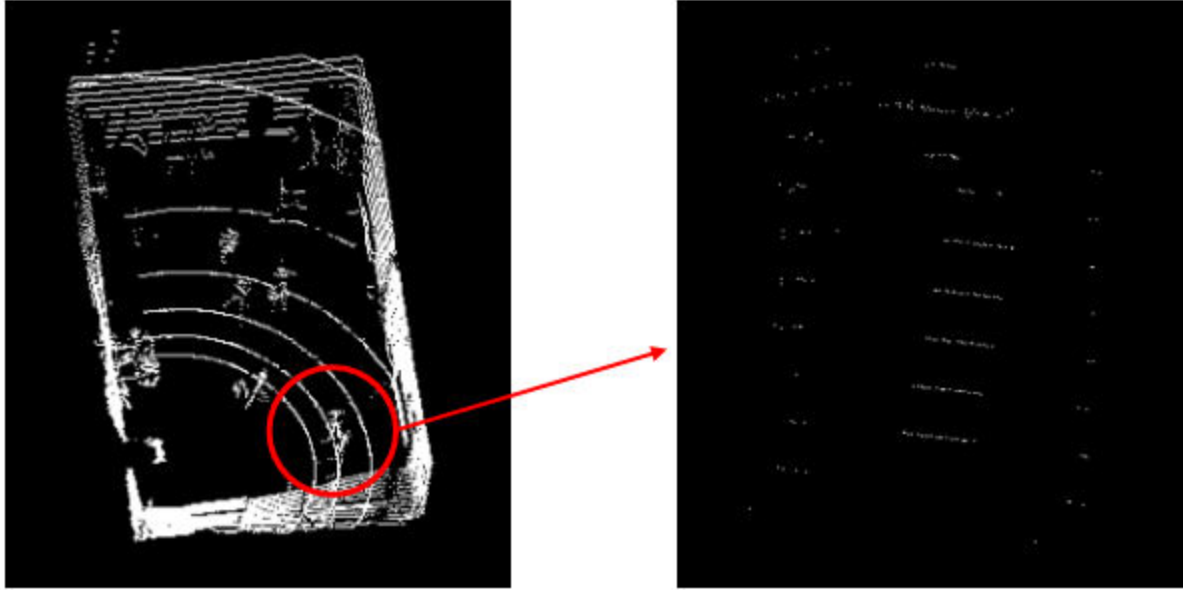


Figure 8: Example of satellite isolation. Left: Raw LIDAR point cloud; Right: Satellite isolated point cloud.

The outlier removal and data downsampling filters described in the main body of the dissertation were also applied to the laboratory data. Examples of this are represented in Figures 9 and 10, respectively.

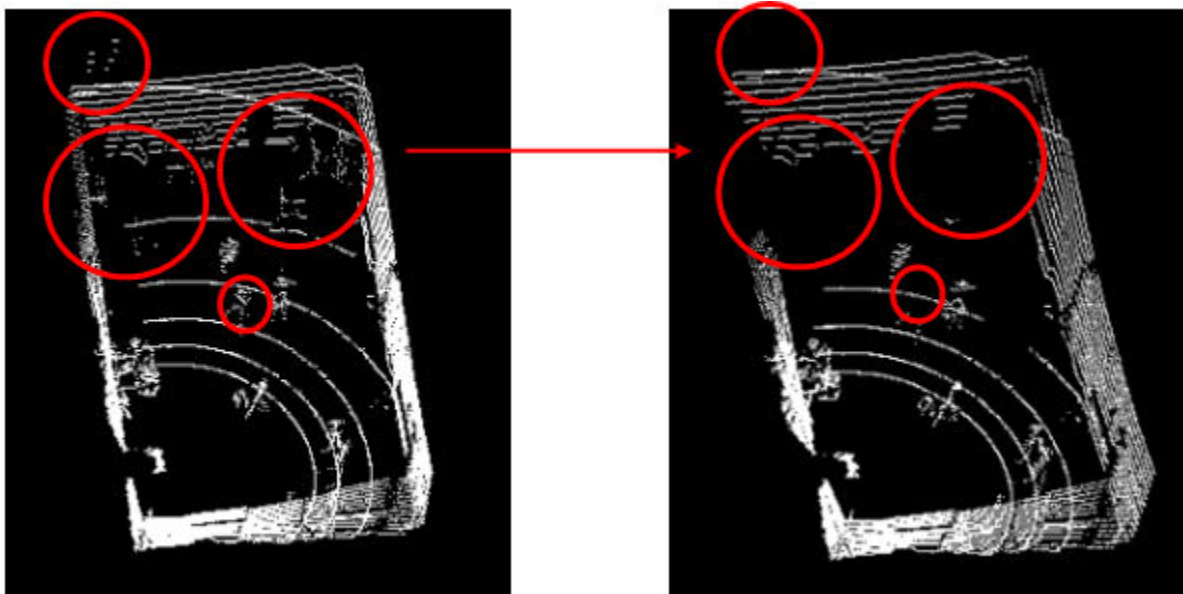


Figure 9: Example of statistical outlier removal: Left - Raw point cloud taken in the laboratory; Right - Point cloud processed to remove outliers (most notable regions are circled in red).

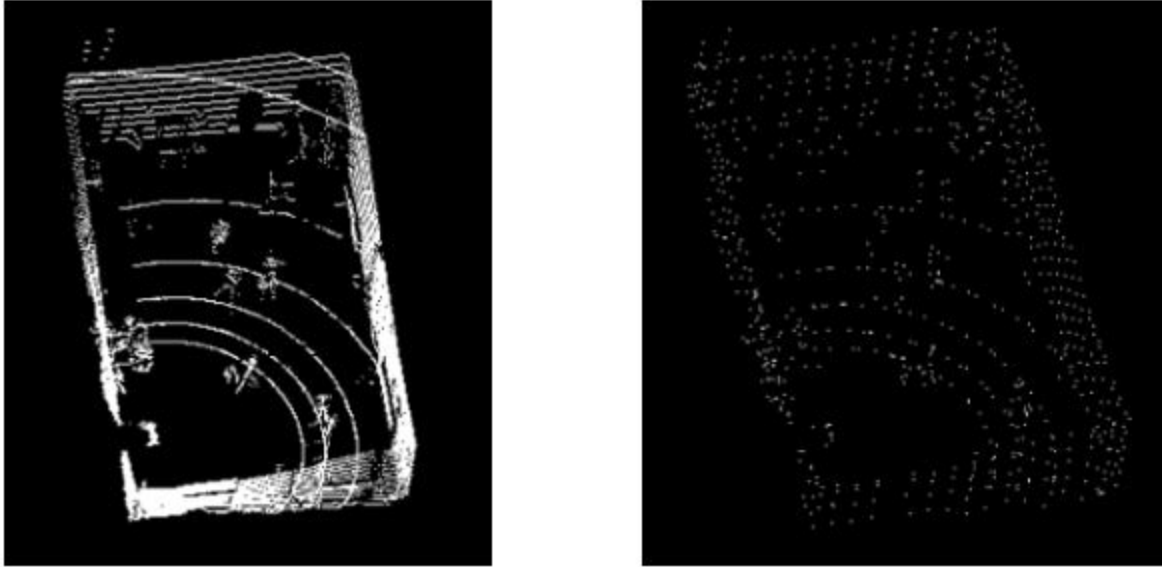


Figure 10: Example of point cloud down sampling. Left: Raw LIDAR point cloud; Right: Down sampled point cloud.

Navigation filters

Finally, due to a blockage in the research and although the data acquisition involved different trajectories around the target satellite, only the performance of the navigation filters on the trajectory 1 data, corresponding to a straight line towards and away from the satellite, was evaluated. To this goal, the trajectory retrieved from the navigation filters was compared to the ground truth recorded by Optitrack. This comparison is showed in Figure 11 for the data representing the whole laboratory and in Figure 12 for the data representing only the isolated satellite. The figures show the results of applying the Generalised ICP navigation filter.

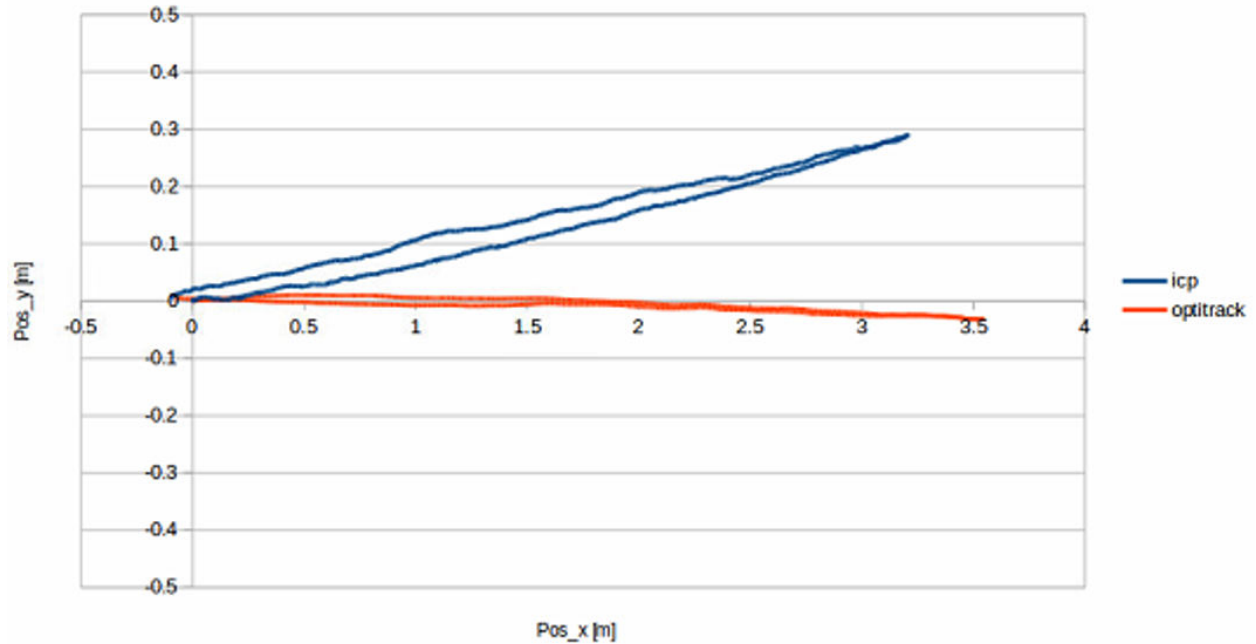


Figure 9: Comparison of the trajectory retrieved from applying the Generalised ICP navigation filter to the LiDAR data and the ground truth recorded by Optitrack. Full laboratory environment.

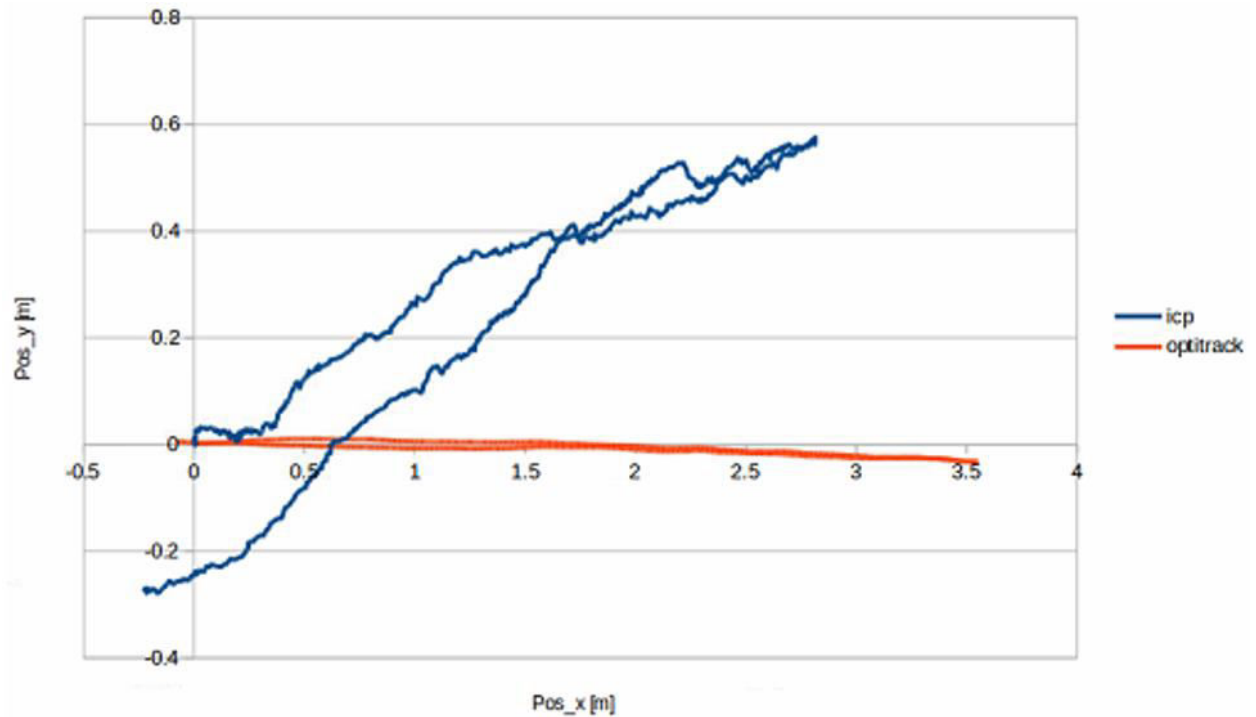


Figure 10: Comparison of the trajectory retrieved from applying the Generalised ICP navigation filter to the LiDAR data and the ground truth recorded by Optitrack. Satellite isolated.

As can be seen in the figures, the comparison of these two trajectories rendered apparent a misalignment between the LiDAR and Optitrack reference systems. Analysing the possible causes, a realisation was made that the Optitrack motion capture system will track the orientation of the LiDAR

relative to that of the rigid body defined in the first frame (i.e. Optitrack is not tracking the LiDAR itself but the IR markers manually located on top of it). However, there is no way to locate the IR markers exactly along the LiDAR axes so that both coordinate systems are aligned. Therefore, any evaluation of the results and computation of errors is not viable between the two frames. This, together with the receipt of simulated data from Thales Alenia Space – France caused the experimental tests to be discontinued. Another argument supporting this decision was the low density of the point cloud after the satellite isolation filter which was not representative of those that the LiDAR will acquire in space.

In the future it will be necessary to solve this problem before proceeding with any further research involving Optitrack and any other reference system for navigation applications. A possible solution might be to define a object that can be tracked by both systems and that can serve as a stepping point between them so to be able to convert between the reference frames and compare the results effectively.