Universität der Bundeswehr München
Fakultät für Luft- und Raumfahrttechnik
Institut für Raumfahrttechnik und Weltraumnutzung

Space Relative Navigation for Autonomous Safe Capture of Non-Cooperative Targets

Harvey Camilo Gómez Martínez

Vollständiger Abdruck der von der Fakultät für Luft- und Raumfahrttechnik der Universität der Bundeswehr München zur Erlangung des akademischen Grades eines

Doktors-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Gutachter:

1. Prof. em. Dr.-Ing. Bernd Eissfeller
2. Univ.-Prof. Dr. Helmut Mayer

Die Dissertation wurde am 14.06.2018 bei der Universität der Bundeswehr München eingereicht und durch die Fakultät für Luft- und Raumfahrttechnik am 07.05.2019 angenommen. Die mündliche Prüfung fand am 05.06.2019 statt.

To P. Alberto, my friend and mentor, whom I will be forever grateful.

Acknowledgements

The last years have been a long but satisfactory stage for my personal and professional development. I am grateful for the time I have been here.

First of all, I would like to thank Prof. Bernd Eissfeller, for giving me the opportunity of developing my research at his institute. His continuous support in different aspects of the project produced humongous enhancements in my investigation, generating better and satisfactory results as those written down in this dissertation.

Secondly, I also want to express my special gratitude to Dr. Hauke Fiedler, who was the Space Situational Awareness research group leader at the German Aerospace Center, and provided me guidance with respect to different aspects related to my project, as well as with diverse issues I had to confront. His support was always valuable and very opportune. Additionally, I am thankful to Dr. Toralf Boge, for the advice given in the initial phase of this venture. His continuous constructive critics made my research to take a second impulse, correcting the path when everything seemed lost. Also, I thank Dr. Oliver Montenbruck for the few, but very rewarding talks, that helped me to improve the quality of my work. I must thank Dr. Heike Benninghoff, for lending me the necessary equipment for the experimental phase of the project.

An special mention of gratitude has to be made to Dr. Gabriele Georgi, who was of great aid and support during the last period of my research. The continuous observations and kind patience produced an enrichment in certain skills in a way I hardly imagined. The latter was possible due to his extensive professionalism, and that was the best lesson I could ever learned.

I am grateful to Munich Aerospace for being the sponsor of my PhD project. All the people related to and involved in this process made easier to bear the load of living in a foreign country. Furthermore, I was pleased of being colleague of

astounding people that influenced me in the best way possible, specially Graciela González, and Gerald Ameres. Also, I would like to thank Prof. Roger Förstner, for his guidance and counsel when required, as well as to Susanne Peters for sharing some of her time, for discussing thoughts, theories, and results for the adequate development of the research topic.

At last, but not least, I want to thank my Mother, my Brother, and my Father, for being understanding and supportive about my decisions during all these years. I strongly believe that the effort will be worth it at the end. And a sincere "thank you" to my girlfriend, who also knows all the process of doing research for a living. She has been of extraordinary support during this time, allowing me to complete my doctorate studies.

Kurzfassung

Die Menge an Weltraummüll ist in den letzten Jahren gestiegen. Die Anzahl inoperativer Objekte nimmt exponentiell zu, was zu Problemen bei aktuellen und zukünftigen Satellitenmissionen in niedrigen und geozentrischen Erdumlaufbahnen führen kann. On-Orbit-Servicing bietet eine Lösung zur Verlängerung der Betriebsdauer von Raumfahrzeugen. Anders ist es mit bereits stillgelegten Satelliten, hochgelegenen oberen Raketenstufen oder Fragmenten und Überresten, die die Sicherheit im Orbit ständig bedrohen. Die aktive Trümmerbeseitigung kann eine einflussreiche Erweiterung der Servicemissionen für Satelliten sein, bei denen ein Chaser-Satellit ein bestimmtes Ziel erfasst und kontrolliert entsorgt, um nicht nur die zukünftige Entstehung von neuen Trümmern zu vermeiden, sondern auch die Trümmerdichte auf bestimmten, stark nachgefragten Umlaufbahnen in bevorstehenden Missionen zu reduzieren. Dies erfordert ein neues Maß an Autonomie in Bezug auf das Rendezvous-Manöver. Neue Herausforderungen gehen daher mit den hohen Anforderungen einher.

Diese Dissertation stellt eine Methodik zur Schätzung der Position und Orientierung eines Raketenkörpers im Orbit, d.h. dem Ziel, unter Verwendung einer Translations- und Rotationsbewegung bezogen auf einen Chaser-Satellit vor. Letzterer hat die Aufgabe, die Dynamik des Ziels für ein sicheres autonomes Rendezvous und die Erfassung anzupassen. Während dieses Manövers verwendet der Chaser eine Time-of-Flight-Kamera, die eine Punktwolke von 3D-Koordinaten erfasst und die Oberfläche des erfassten Ziels kartiert. Sobald das System das Ziel identifiziert hat, initialisiert es die relative Position und Ausrichtung des Chasers zum Ziel. Nach der Initialisierung ermöglicht ein Tracking-Verfahren dem System, den Verlauf der Pose des Ziels zwischen den Frames zu erfassen. Die vorgeschlagenen Algorithmen werden anhand von simulierten Punktwolken, die mit einem CAD-Modell des Ziels und den Kameraspezifikationen des PMD CamCube 3.0 erzeugt wurden, bewertet.

Zusätzlich wurde ein skaliertes, gedrucktes Modell mit einem tatsächlichen Sensor verwendet, um reale Daten zu erfassen und die Leistung der Routinen zu beurteilen.

Abstract

The number of space debris population has risen during the last years. The amount of inoperative objects is exponentially increasing, and this may cause problems in current and future satellite missions in both Low and Geocentric Earth Orbits. On-Orbit Servicing is rising as a solution for extending the operative life of spacecraft. But the situation changes with already defunct satellites, high-altitude upper rocket stages, or fragments and leftovers, which are continuously threatening the safety in orbit. Active Debris Removal can be an influential expansion of the satellite's servicing missions, where a chaser satellite will capture a designated target, and discard it in a controlled manner not only to avoid future production of new debris, but also to reduce the population on certain high demand orbits in forthcoming missions. This requires new levels of autonomy with respect to the rendezvous maneuver. Hence, new challenges go along with those demanding requirements.

This dissertation presents a methodology for estimating the position and orientation of a rocket body in orbit, i.e. the target, under a translational and rotational motion with respect to a chaser spacecraft. The latter has the task of matching the target's dynamics for a safe autonomous rendezvous and capture. During this maneuver, the chaser would employ a Time-of-Flight camera that acquires a point cloud of 3D coordinates, mapping the sensed target's surface. Once the system identifies the objective, it initializes the chaser-to-target relative position and orientation. After initialization, a tracking procedure enables the system to sense the evolution of the target's pose between frames. The proposed algorithms are evaluated using simulated point clouds, generated with a CAD model of the target, and the PMD CamCube 3.0 camera specifications. Additionally, a scaled printed model was used with an actual sensor to capture real data and assess the routines performance.

Contents

List of Figures

List of Tables

Nomenclature

Roman Symbols

$\vec{n}$	Surface's normal at a query point
P_i	ith-point in point cloud
$\mathcal{P}$	Point cloud set

Acronyms / Abbreviations

ADR	Active Debris Removal
CAD	Computer-Aided Design
CCD	Charge-Coupled Device
CMOS	Complementary Metal-Oxide-Semiconductor
CVFH	Clustered Viewpoint Feature Histogram
DART	Demonstration of Autonomous Rendezvous Technology
DLR	Deutsches Zentrum für Luft- und Raumfahrt - German Aerospace Center
DOF	Degrees-of-Freedom
EKF	Extended Kalman Filter
EOL	End-Of-Life
EPOS	European Proximity Operations Simulator
ESA	European Space Agency

ETS-VII	Engineering Test Satellite-VII
EVA	Extra-Vehicular Activities
FOV	Field of View
FPFH	Fast Point Feature Histogram
GEO	Geostationary Earth Orbit
HST	Hubble Space Telescope
ICP	Iterative Closest Point
ISS	International Space Station
JAXA	Japan Aerospace Exploration Agency
KF	Kalman Filter
LED	Light-emitting diode
LEO	Low Earth Orbit
LOS	Line-of-sight
MEO	Medium Earth Orbit
NASA	National Aeronautics and Space Administration
NIR	Near-Infrared
OOS	On-Orbit Servicing
OUR-CVFH	Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram
PFH	Point Feature Histogram
PMD	Photonic Mixer Device
RANSAC	Random Sample Consensus
SfM	Structure from Motion

SL-8	Name designated to a family of rocket bodies cataloged as space debris
SLAM	Simultaneous Location and Mapping
SVD	Singular Value Decomposition
ToF	Time-of-Flight
VFH	Viewpoint Feature Histogram
XSS-11	Experimental Satellite System-11

1 Introduction

1.1 Motivation

Different facts have characterized the space exploration since the launch of Sputnik in 1957, and the subsequent "Space Race" during the following decade, which ended with the landing of the man on the Moon. Many technological milestones developed during the last half of the 20th century until today have made a great impact in the development of the robotics. The latter was extremely important in order to explore remote locations where the human being is not capable of doing so by its own means. But also new requirements in Earth observation, remote sensing, and satellite communications have arisen side by side with those technologies. As a result, the population of the Low Earth Orbit (LEO) and the Geostationary Earth Orbit (GEO) has considerably increased during the last decades. There are also some population in the Medium Earth Orbit (MEO). Some of those satellites are floating around without control, being a threat for current operational missions. Also rockets' leftovers, small pieces of those objects, and tools lost by astronauts during spacewalks are also a menace for the operative satellites.

Those residuals are commonly named "space debris", which are continuously increasing based on the phenomenon identified as "Kessler syndrome", defined as the tendency of exponential increase of the number of objects due to collisions among themselves (Kessler and Cour-Palais, 1978). Therefore, a remediation for this problem enforced the necessity of applying space robotics in how to reach those objects and dispose them properly. Before thinking about the means of capture/collection of those bodies, two of the more demanding issues to solve are the identification of the object as the designated target, as well as the estimation of its relative position and orientation with respect to a chaser satellite.

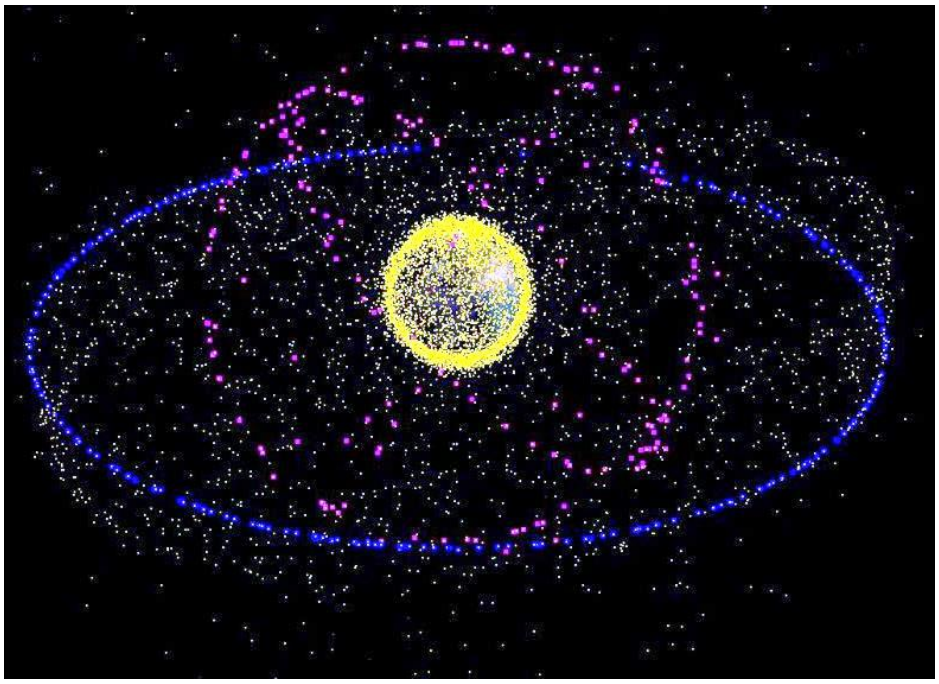


Figure 1.1. Simulation of the space debris belt on LEO–yellow, MEO–magenta, and GEO–blue. The objects are not at scale. (*Credit: DLR*)

1.1.1 Space Debris

As said above, the first man-made object that orbited the Earth was the Sputnik-1, on October 4th, 1957. Since then, different missions had been launched for diverse purposes, e.g. exploration, communications, Earth observation, meteorology, navigation systems, et cetera. As a result of those missions, some objects remained on orbit because of their specific function during the different phases of the spacecraft operation, and they were not properly disposed. For instance, upper stage rockets for the high-altitude orbit insertions, or satellites after fulfilling their mission. It is also possible to find satellite-to-launcher adapters, upper stage fairings, or diverse clamps/bands/retainers for movable devices, that are increasing the number of space debris elements.

On the other hand, unintentional items –tools or gloves of the astronauts–, leftovers of the launcher operation –particle products of solid motor combustion, or diverse internal liquids leakage–, and coating/painting degradation of both the satellite's and upper stage's surfaces are other sources of space debris. Additionally, the sudden explosion of spacecraft and rocket stages is also an important source

of debris, mainly produced by the abrupt change in temperature on orbit, making the unused fuel to ignite inside the satellites and upper stages. Therefore, the space debris have been cataloged based on their function/operation, into three groups mainly:

- Fragments
- Payloads
- Rocket bodies

Because of more than 60 years of space operations, some space regions have had more exploitation than others due to different factors, like better illumination, desired orbital period, among others. For instance, most of the operative spacecraft are located in LEO, because they are close enough for remote sensing and Earth observation missions, and those satellites are protected from the cosmic rays thanks to the Earth's magnetic field. Additionally, some of those orbits at LEO have certain characteristics that make them desirable for specific missions, e.g. Sun-Synchronous orbits maintain the same sun illumination conditions over the year, making this fact very useful for Earth observation using cameras. These orbits have inclinations greater than $90°$, with the most used altitude between 900 and $1500\mathrm{km}$ (National Research Council, 1995; United Nations, 1999). On the other hand, spacecraft at GEO have an orbital period of approximately $23\mathrm{h}$, $56\mathrm{min}$, and $4\mathrm{s}$, or the same as a mean sidereal day. This ensures that the spacecraft remains around the same longitude during the whole time. This fact is useful for communications, because satellites seem to be fixed in sky for ground antennas.

Hence, the demand of having a catalog that describes the different type of space debris objects arises as new spacecraft are launched year after year. Most of the data is observed by the US Space Surveillance Network, maintained in their catalog through data collection, based on ground radars and telescopes network. The latter watches the sky, detecting and tracking most of the space objects. It is not possible to watch all of them due to technical limitations, since the average size of observable debris is $5 - 10\mathrm{cm}$ in LEO, and $30\mathrm{cm}$ to $1\mathrm{m}$ in GEO (Klinkrad, 2006). Figure 1.2 shows the orbital spatial distribution of the cataloged elements.

For those objects not-observable from Earth, the measurements have been made in situ by optical means, or even some evidence has been taken from impacts of small

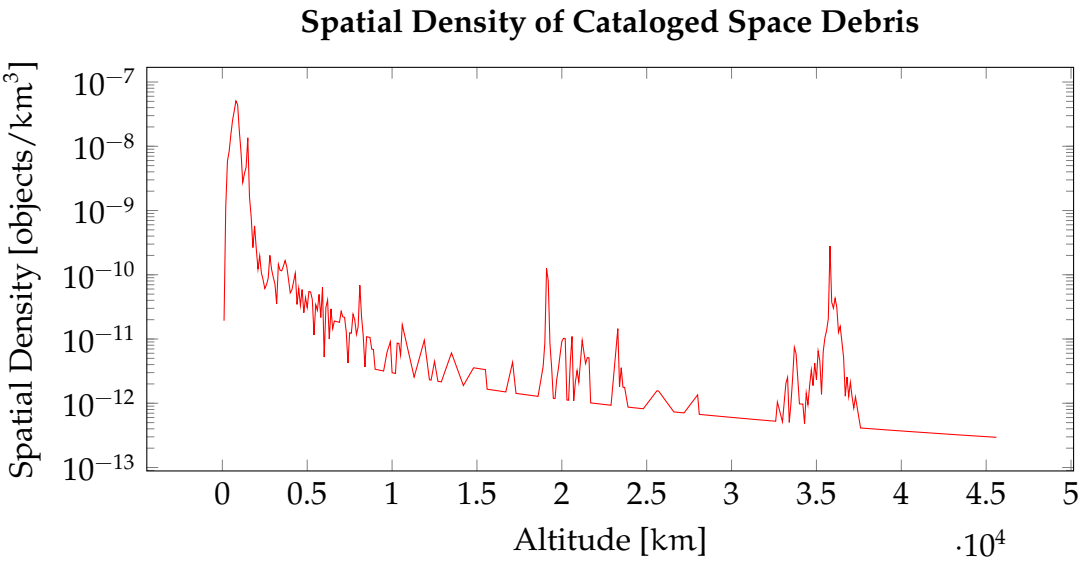


Figure 1.2. Spatial Density of Cataloged Space Debris.
(*Data source:* `www.space-track.org`)

fragments on diverse spacecraft. Those recordings allow to reproduce environmental models of the debris population. Thus, another important classification topic for the space debris is their size, along with the estimated mass. This arrangement can be seen in Table 1.1 (National Research Council, 1995).

Table 1.1. Debris Size Classification

Size	Diameter	Mass
Small	$< 1\text{mm}$	$< 1\text{mg}$
Medium	$1\text{mm} - 10\text{cm}$	$1\text{mg} - 1\text{kg}$
Large	$> 10\text{cm}$	$> 1\text{kg}$

The altitude is a determinant factor for space debris. For instance, the velocity needed for an object to stay in orbit is about $6.9 - 7.8\text{km/s}$, depending on the LEO height, and an average velocity of 3.1km/s for satellites in GEO. Then, a fragment, whose speed can be 10 times the speed of a bullet –for LEO–, can easily perforate any part of a satellite due to the high momentum of each piece. Additionally, when the mass of the target increases, the kinetic energy also rises, making the threat greater. As an example, an object of 500kg at a speed of about 7.8km/s can have a kinetic

energy of about 1.521×10^{10}J. The latter is comparable to about 3.6 times the energy released by the explosion of 1t of trinitrotoluene, i.e. 3.6 megatons, approximately.

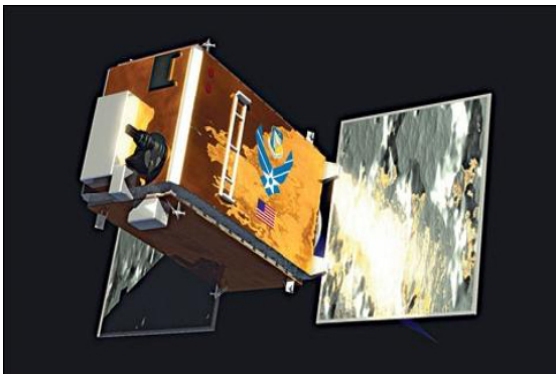
However, not only the orbital velocity and the mass are critical for an impact, but also the relative velocity two objects may have. The collision velocities can vary from 0km/s if both objects are in the same orbit, to twice the orbital velocity if the objects are facing each other. A special mention has to be made when the impact is produced by an object in an elliptical orbit. The latter has the highest velocity at the perigee, which will be faster than the local circular orbital velocity. Otherwise, the lowest speed is located at the apogee, which will be lower than the local circular orbital velocity. Thus, even higher relative velocities could be achieved. Additionally, the intersection angle of the orbits has an influence in the collision velocity, e.g. if the angle between two orbits is greater than 60°, the relative velocity would be larger than the local orbital velocity (National Research Council, 1995).

1.1.2 Autonomous On-Orbit Servicing

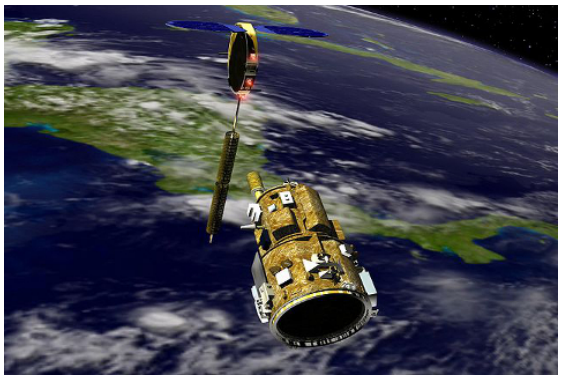
On-Orbit Servicing (OOS) is defined as the capability of a master spacecraft to rendezvous another operative spacecraft, and perform certain tasks on the latter that may correct and/or enhance its conditions, in order to have an improved operation (NASA, 2010). OOS surged as a necessity of extending the End-of-Life (EOL) of the satellites around the Earth by refueling, replacing power batteries, or adapting/changing payloads for mission extension. Servicing on spacecrafts has been done in past, specially on space stations like Skylab or the International Space Station (ISS), and on the Hubble Space Telescope (HST).

One particular factor in the samples described above refers that the astronauts are who perform Extra-Vehicular Activities (EVA), i.e. spacewalks, in order to repair and/or update the hardware of the serviced spacecraft. Sometimes those tasks can be really demanding, or even dangerous for the people who perform these activities. Furthermore, the prevalence of factors like safety and integrity for both servicer and under-servicing spacecraft granted improvements in autonomous proximity operations. For instance, because of the high orbital velocities both spacecraft have, the rendezvous requires a very precise operation in order to keep a constant distance in between. For that reason, the search of a bigger autonomy level in the process was imperative. In addition to this, the rendezvous maneuver can be limited due

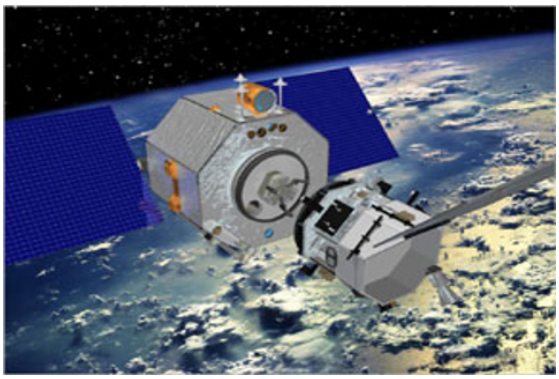
(a) ETS-VII



(b) XSS-11



(c) DART



(d) Orbital Express

Figure 1.3. Missions of autonomous rendezvous technology demonstration.

to short communication windows from ground station. Hence, the mission could take additional resources that could be reduced, if the rendezvous is made in an autonomous way.

Because of all the described reasons, some technology demonstrators like the missions Engineering Test Satellite VII (ETS-VII), Experimental Satellite System-11 (XSS-11), Demonstration of Autonomous Rendezvous Technology (DART), and Orbital Express (Woffinden and Geller, 2007), tested the possibility of performing servicing tasks without astronauts involved in the process. In these missions, either the target provided its own position and attitude to the chaser, or ad hoc visual markers on the target's surface enabled the chaser to autonomously obtain the relative pose (Christian et al., 2012). Some assessments were successful, while others were not. However, the average outcome of those tests was the high probability of performing autonomous OOS, under certain operative conditions.

1.1.3 Active Debris Removal

Before the idea of servicing, the spacecraft was considered to be orbiting around the Earth after the completion of its mission. Those "defunct" satellites were considered junk, or debris. Some of them returned to Earth burning up due to the friction with the atmosphere at reentry, but exist others that could survive the fall and become a threat for the population. Furthermore, when a satellite is near to its EOL, it can be put into a graveyard orbit, where the influence of atmosphere is nearly minimum and can be there for tens of decades, suffering almost no decay in their altitude. Unfortunately, there are other objects that exist in between, and they can become a threat for operative spacecraft and satellites, e.g. the ISS, which have performed evasive maneuvers to ensure both the integrity of its structure and the safety of its occupants.

Although space surveillance catalogs exist for tracking the space debris, the number of elements is continuously increasing, making harder to ensure the integrity of the operative satellites. For that reason, like the OOS, another servicing necessity has emerged in order to reduce the population of space junk: Active Debris Removal (ADR). ADR is defined as the process of sending a chaser satellite to capture a certain target, and dispose it properly, without producing additional debris during the process. Both ADR and OOS refer to rendezvous to an object on orbit, but OOS has the advantage that the target would still be operative and provide some information about its status. In this scenario, the target is *cooperative* for the OOS. On contrast, space debris are non-operative devices that have finished their duty time and were not designed to be recovered after EOL. For this reason, the debris are named *non-cooperative* targets.

This non cooperativeness increases the level of complexity to the autonomous proximity operations, because of the absence of knowledge about the dynamics of the target. Additionally, due to the lack of orbital control of the debris, the latter may be tumbling. Hence, the change of the target's attitude makes even harder on how to define the approach method to capture the debris in a safe way for both chaser and target. For that reason, before the capture of the object, it is important to perform a trade-off between the possible methods for detecting and tracking the debris.

Furthermore, all the process should be done in an autonomous way, based on the possible interruption of the communication links between the ground station

and the chaser satellite. In the case of LEO, the orbital period indicates how many minutes a spacecraft could pass through the coverage area of a ground antenna. If a critical maneuver –e.g. capturing the target using a robotic arm– is being performed remotely from the station, and the communication link is broken during that precise moment, the risk of failing the mission is evident due to lack of information at the current epoch. Moreover, a communication network could be set using multiple ground stations and/or relay communication satellites at higher altitudes. However, the latter could induce latency in the communication signal. This produce a mismatch between the data that the operator is observing, and the current measurements from the on-board sensors.

The critical step is the selection of an object as a target for a possible ADR mission. It is necessary to take into account different aspects to consider a space debris object as a candidate for removal:

- Orbit, i.e. the height and the inclination plus the population of elements in a specific region.
- Collision probability, or how big the chances are for producing new debris due to impact and/or explosion.
- Size, which determine the quantity of fragments, if the object breaks apart.
- Mass, that implies the amount of matter of new fragments, if they are generated.

Based on the previous considerations, it implies that the suitable candidates to be removed are the rocket bodies, due to their big size and mass, additional to the high impact probability with other orbiting debris. Furthermore, rocket bodies are a source of new debris objects due to explosion risks, result of unused fuel inside the propulsion system, as described in Section 1.1.1. Based on the previous premises, the most promising targets to be actively removed are the SL-8 Rocket Bodies (Peters et al., 2015). This nomenclature refers to the second stage of the Russian launcher Cosmos-3M, whose Computer-Aided Design (CAD) model is shown in Figure 1.4.

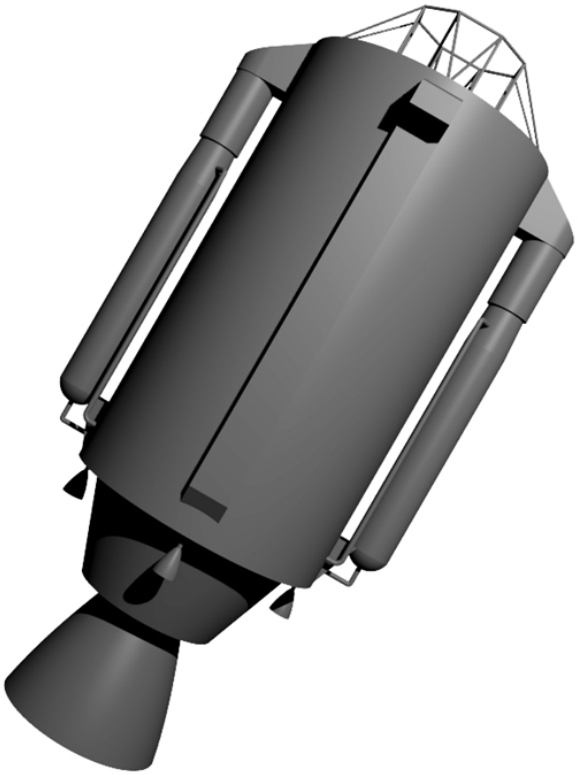


Figure 1.4. CAD model of the Cosmos-3M upper stage.

1.2 Visual Navigation Methods for Space Rendezvous

When two spacecraft are going to encounter, the rendezvous maneuver is divided in 4 main phases, based on the distance between them. They are: phasing, far range, close range, and final approach (Fehse, 2003). In case on ADR, the same rendezvous scheme can be hold. However, the sensors selected for this kind of mission can differ to those used in OOS. The latter is based on the nature of the targets of being non-cooperative.

The trade-off of the sensors are based on their capabilities and what is expected as a result. The required outcomes are commonly the distance between the target and the chaser satellite, and the attitude of the target with respect to the chaser reference frame. Thus, the suitable option for achieving those outcomes is by using visual navigation sensors. The main advantage of this type of devices is the possibility of *detecting* an object, i.e. *see* the shape and appearance of it, as well as *tracking* it, i.e. *following* its movements during a certain lapse.

As the ranging is one of the unknowns to solve, the system requires a setup for such purpose. There are *active* and *passive* sensors that can be useful for solving this issue. Passive sensors, or cameras, detect a portion of the wavelength emitted or reflected by the objects in order to generate images of them, without any additional action than recording consecutive frames. By itself, a camera is not able to measure distances. However, some improvements has been made in order to record such data.

For instance, the stereo triangulation makes possible the extraction of depth in a scene, with the use two or more cameras and a well-known baseline between them. Otherwise, the range information can also be obtained by using a single camera when one of the two actors involved is moving, i.e. either the camera or the observed object. This method is referred as structure from motion (SfM) (Bonin-Font et al., 2008). Although the camera is a cheap sensor and its mass is low, the requirement of having textured surfaces, which are used for detecting features and reconstruct the 3D points of the surface, makes difficult its implementation. Additionally, the illumination conditions on orbit change quite fast, making hard the proper algorithm operation. Finally, the scale ambiguity using cameras makes hard to distinguish between a big target that is far from the sensor and a small object that is very close. The latter will be discussed in more detail in Section 2.2.1.

On the other hand, active sensors help to overcome some of the drawbacks described before for the cameras. For instance, the light detection and ranging sensor, or lidar, emits their own illumination with lasers in a certain wavelength range (e.g. $500 - 1600nm$), being less susceptible to variation in the lighting conditions. The great advantage of lidar is related to their capability of measuring the distance between the target and the sensor, as well as the possibility of reproducing its surface based on the range data acquired. Each range recording is a point in the 3D Euclidean space of the sensor reference frame. Hence the set of all those points after acquisition are commonly called "point clouds". Otherwise, the accuracy of those range measurements can be reduced by the reflectivity properties of the object, and the radiation emission carries more power consumption to the system.

In between, different variants of the lidar exist based on the way these devices capture the 3D information. First of all, the scanning lidar uses a laser beam pointed to the requested scene/object through a steerable mirror, which directs the laser beam. For each step the mirror has, a range measurement is made. The most

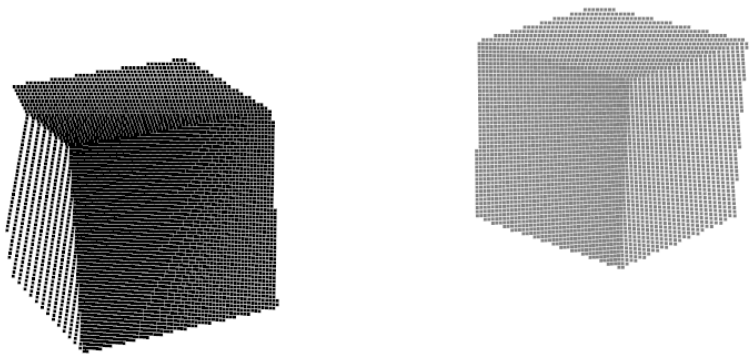


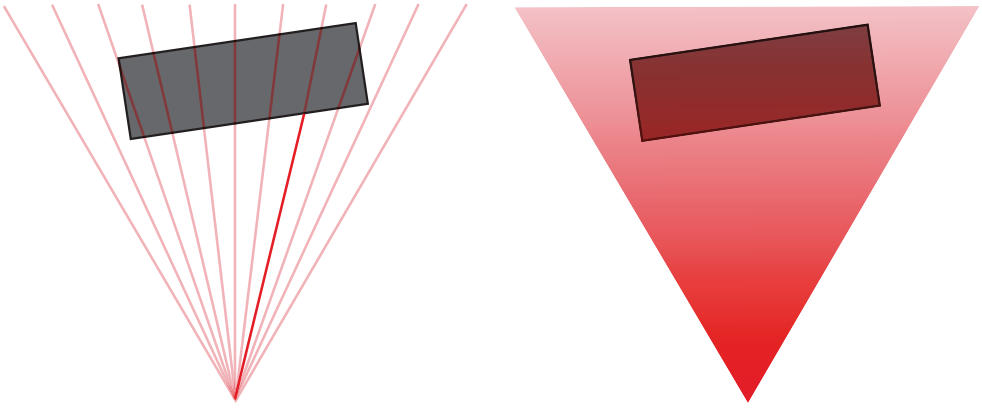
Figure 1.5. Example of point clouds.

common types of this kind of lidar are the single-line scanning and the bidirectional pattern scanning. The former steers the mirror in a way that all the measurements lies on a plane (see Figure 1.6a). On the contrary, the latter has the possibility of moving in 2 directions instead of one, obtaining a more descriptive information of the scene.

Although the accuracy of lidar is in order of centimeters, and it can cope with adverse illumination conditions, the use of gimbals and mirrors to redirect the laser beam is a kind of drawback for the proposed ADR mission, based on the weight and power consumption of the sensor. Additionally, the target is supposed to be tumbling free on orbit. This dynamics may produce blur motion on the data, due to the time required by the lidar to scan the scene.

Otherwise, the second variant of the lidar is called flash lidar, in which the laser beam is optically diverged to produce a whole illumination of the scene/object in a single laser shot without the use of mechanical devices like those used in the scanning lidar (see Figure 1.6b). The sensor has a pixel array similar to those used in passive cameras, which capture the reflected light. Each pixel is capable to record the range by measuring the time required by the emitted light since it was shot and the reflection returns to the sensor. Therefore, the flash lidar can produce the 3D representation of the surface with the range information in each pixel of the sensor.

Regarding the drawbacks, the flash lidar also has some of consideration. First of all, the sensor captures the light in the field of view, coming from the reflected light as well as from other illumination sources. This may expose the sensor to saturation, leading a bad range measurement. Secondly, the operational range is lower than that



(a) Scanning lidar (b) Flash lidar

Figure 1.6. Comparison of the operation between a scanning lidar and a flash lidar.

from the scanning lidar, because of the diffraction of the laser beam, which produces a light with a lower intensity (Pollini et al., 2011).

During the last decade, a new sensor has capture the attention of applied robotics, and it is called Time-of-Flight (ToF) camera. As its name indicates, it also measures the range towards an object based on the time the photons "flight" from the camera, impact on the target and return to the sensor. In the sense of "camera", it is capable of producing not only a 3D surface representation, but also a 2D gray-scale image of the object based on the measurement of the amplitude of the reflected light in one single shot (Ringbeck and Hagebeuker, 2007). The illumination source is an array of light-emitting diodes (LED), that have their operational wavelength in the near-infrared (NIR) section of the electromagnetic spectrum, i.e. similar to that used in lidar. Because of the low power consumption for emitting light, low weight in comparison to the laser counterparts, as well as the similar operation principle to the flash lidar, the ToF cameras are considered to be candidates for proximity operations, despite of the lack of qualification for space flight.

However, the ToF camera also has the similar disadvantages to those related to the flash lidar. The use of led arrays instead of laser reduces the operative range measurement. Also the multi-path errors, i.e. illumination of other sources than the object reflected light, are common in ToF camera. On the other side, the capability of having its own illumination source, as well as producing 3D measurements, make this sensor a good option for full darkness conditions. For instance, if the ADR is performed in LEO, the orbit may have a shadowed portion given by the Earth

and the duration of this eclipse is about one third of the orbital period. This can be advantageous for estimating the position and orientation of the space debris.

1.3 Related Work

The main technological demonstration missions about autonomous rendezvous have been already addressed in Section 1.1.2. In those cases, the estimation of the position and orientation, also called pose, has been done in a cooperative way, through the use of markers or reflectors on the target spacecraft, in order to identify them and subsequently tracking. Similarly, different type of lidar has been used in relative navigation on orbit (Christian and Cryan, 2013), being some of them related to the demonstrators, as well as flying experiments aboard the Space Shuttle.

Moreover, additional research has been made in laboratory in order to validate new improvements with respect to the goals achieved by the flown missions, using different approaches for points clouds manipulation as well as types of sensors. In the laboratory experiments (Aghili, 2010; Aghili and Parsa, 2009; Ruel et al., 2008), scanning lidar were used for the state estimation, and then the outcome was used for proper autonomous rendezvous. Basically, the motion of the target spacecraft is always assumed to be moving slowly relative to the time frame between scans. The last statement was done because the scanning time of the lidar may be slower than the spin motion of the target. If this happens, the collected range data will suffer of "smearing", which can be compared to the blur in common photograph images. The data smearing produces a bad geometry representation of the target, if it is not corrected. Hence, having the low rotation speed assumption, it makes the solution to be more restrictive in comparison to the motion of a non-cooperative, tumbling debris. Furthermore, The German Aerospace Center (DLR - Deutsches Zentrum für Luft- und Raumfahrt) has been performing OOS studies for the last meters of rendezvous using visual navigation (Boge et al., 2013; Sellmaier et al., 2010). For that purpose, the orbital conditions are simulated on the European Proximity Operations Simulator (EPOS) (see Figure 1.7). Here, due to the purpose of OOS, it inferred that the relative motion between the two spacecraft is quite slow.

With respect to the ADR, different methods has been studied in order to detect and track the space debris (Bonnal et al., 2013), as well as the means for capturing them (Shan et al., 2016). Here, a chaser platform was assumed to be equipped with a

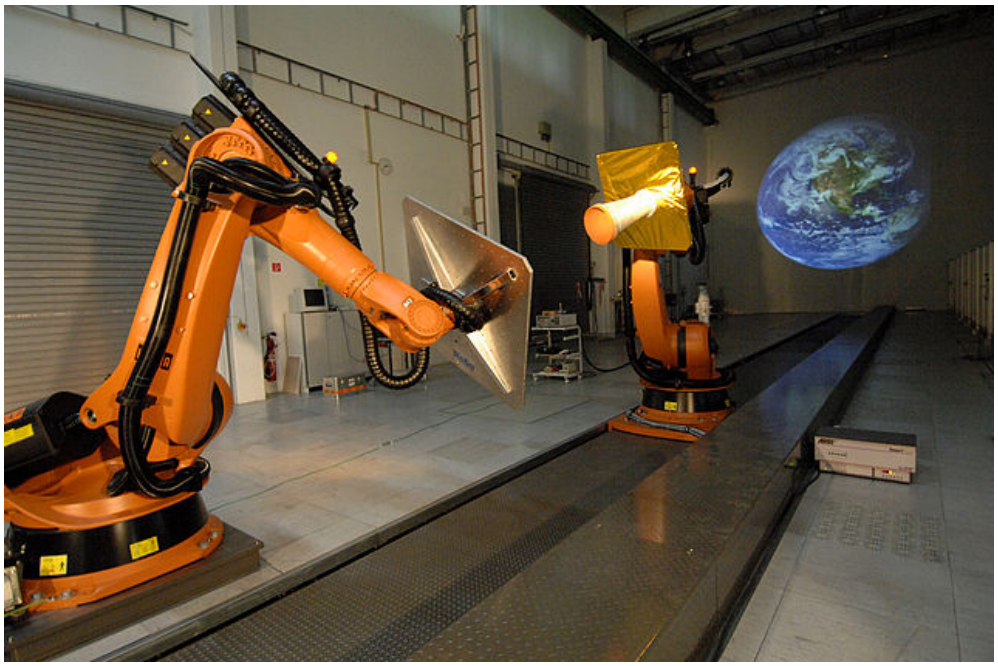


Figure 1.7. EPOS in the facilities of DLR in Oberpfaffenhofen, near Munich (*Credit: DLR*)

robotic arm (Felicetti et al., 2016). Recently, a technology demonstration mission from the European Union has been announced with the name of "Removedebris" (Lappas et al., 2014). This mission has been designed in order to perform three different experiments on orbit using three cubesats as mock-ups, respectively. Two of them are regarding to capture methods, specifically a net and a harpoon. The remaining dummy cubesat will be subject of navigation algorithms validation for different rendezvous maneuvers using a standard visual camera and a flash lidar. All the data acquired during this mission will be processed on-ground, meaning that the rendezvous operation will not be autonomous. Even though, this would be one of the first missions focused on validating methods for ADR.

Although one might think otherwise, the main space agencies have focused their efforts in different directions, and not all of them are related to ADR. For instance, the National Aeronautics and Space Administration (NASA) and its Orbital Debris Program Office (Liou, 2017) has been developing their research activities towards observations from ground in order to characterize the orbital debris population. Additionally, this department is developing particle impact detection technologies for in situ measurements. Finally, the Office also develops, maintains, and updates space debris environment models. In the case of the Roscosmos State Corporation for

Space Activities, commonly known only as "Roscosmos", the efforts for space debris mitigation are now focused on adopting new policies for future missions, which may reduce the augmentation of the space debris population. They are also planning new methods and procedures for proper spacecraft disposal after EOL (Makarov et al., 2017). But, surprisingly, both American and Russian space agencies are not promoting any ADR activity by themselves.

On the contrary side, the Japan Aerospace Exploration Agency (JAXA) has been working recently in a concept for ADR (Kawamoto et al., 2017), where they are assessing the possible sensors to be used –cameras and lidar–, as well as the trajectory scheme, and the means for capture. For the disposal method, they are evaluating an electrodynamic tether, which is a long conducting wire that generates energy by passing through the magnetic field of a planet –the Earth in case of ADR– . Then, the tether can be used as a thruster, ejecting electrons generated by the magnetic field, as well as those that float freely in the ionosphere, also captured by the tether. Hence, the debris has now a propulsion device to enter in a controlled way into the atmosphere. JAXA is proposing to make an on-orbit demonstration around 2020.

At last, but not least, the European Space Agency (ESA) is determined in removing a single large ESA-owned space debris from LEO. The e.deorbit mission is the answer to achieve that goal in the next years (Biesbroek et al., 2017). This mission is the result of the necessity of disposing in a proper way the defunct ENVISAT, which is tumbling in an uncontrolled way, and with a mass of around 8 tons. ESA has been evaluating the different sensors –most of them both passive and active cameras–, methods of capture, and guidance procedures to perform a safe ADR through simulations and laboratory tests. ESA plans to demonstrate the technical advances on orbit and the possible disadvantages in 2023.

1.4 Contributions

This thesis presents the development of quasi-real time algorithms for the autonomous estimation of the 6 Degrees-of-Freedom (DOF) of a non-cooperative, tumbling rocket body in orbit, previously selected as target, using a ToF camera.

1. The algorithms use individual, consecutive point clouds, fundamental for the two main phases of the estimation:
   (a) Initialization: to estimate the target position and orientation for an initial state.
   (b) Tracking: to preserve the pose estimation over time due to the motion of the target.
2. The performance of the algorithms was evaluated in two conditions:
   (a) Use of simulated point cloud data, based on the characteristics of a commercial ToF camera.
   (b) Use of real point cloud data, operating a ToF camera on a 1:10 scaled, 3D printed model of the rocket body.

1.5 Outline

The current chapter has presented the motivation for this research, and an overview of the state of the art for relative navigation to cooperative targets on orbit. Additionally, the contributions of this thesis to the field but for non-cooperative targets.

Chapter 2 details the theoretical foundation image formation, as well as the ToF camera operation, and how the point clouds are produced. It also describes the fundamental point cloud operations.

Chapter 3 defines point cloud feature descriptors, and the point cloud registration problem.

Chapter 4 presents in detail the problem of position and attitude estimation of a free-floating object, and the proposed approach of this research work.

Chapter 5 describes the validation of the algorithms with both simulated and real point cloud data, in order to show the performance of the solution.

Chapter 6 summarizes the thesis, showing the conclusions of the work, and proposing further recommendations for future research.

2 Image Theory

This chapter presents how it is possible to measure range using a camera and its illumination source, under certain operation conditions. This technique allows not only to measure the distance between the agent and the target, but also to have a visual representation of the object in both 2D and 3D. Thanks to the last outcome –the 3D information–, robotics has greatly enhanced, because it allowed to autonomous agents to interact with the environment, with more safety and better performance. Of course, different range measurement methods have existed before by using radio-frequency, i.e. radar, or ultrasound, i.e. sonar. But those techniques have some drawbacks that may affect the operation of an autonomous system. For instance, the use of ultrasound in vacuum is impossible, because of the absence of a transmission medium. Contrary to the radar that can work in different means, the most basic arrangement can only detect objects in one dimension. Otherwise, active imaging allows not only to measure the range, but also to identify the object and check its condition, if required.

2.1 Time-of-Flight theory

Active 3D imaging is based on the Time-of-Flight (ToF) principle: it employs the speed of light of an emitted signal to calculate the range. An array of LEDs produces NIR light, which is modulated by a emitted signal of lower frequency (typically a few tens of MHz). This light hits on a scene, and the portion of the reflected light is captured by a receiver sensor that records both brightness and emitted-to-received phase shift (see Figure 2.1). The latter enables to extract the range ρ between the

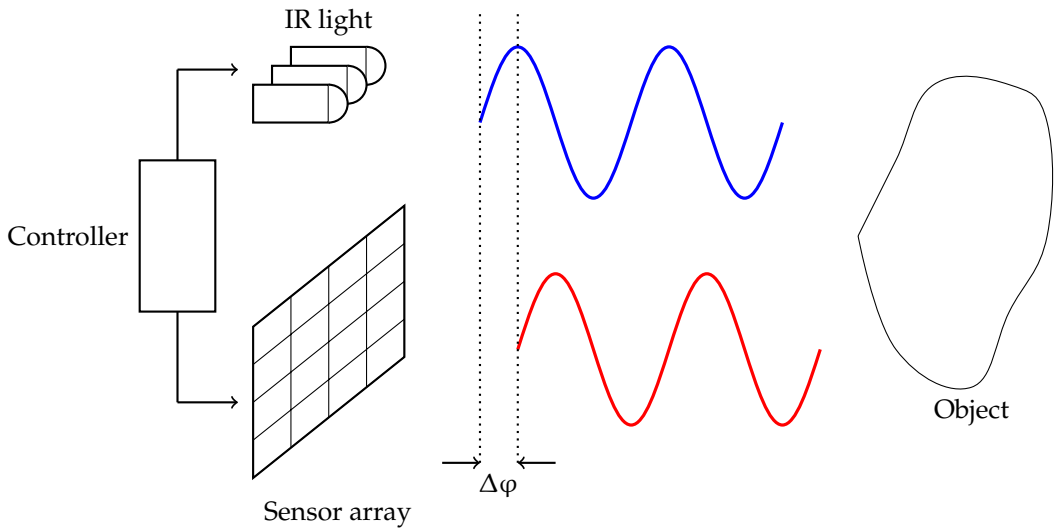


Figure 2.1. Principle of operation of the ToF camera. The phase difference between the emitted (blue) and received signal (red) helps to measure the distance from the sensor to the object.

camera and the observed scene, based on the following equation:

$$\rho = \frac{c}{2f}\frac{\Delta\varphi}{2\pi} \tag{2.1}$$

being c the speed of light, f the signal modulation frequency, and $\Delta\varphi$ the signal phase difference (Hansard et al., 2013).

2.1.1 Photonic Mixer Device

Different type of sensors are used for capturing the intensity of light in order to generate images. Common image processing systems are based on Charge-Coupled Device (CCD) or Complementary Metal–Oxide–Semiconductor (CMOS) sensor, that allow to project the information of 3D world into a 2D image. Because of the latter, a new technology has been produced in order to obtain 3D information: Photonic Mixer Device (PMD) (Ringbeck and Hagebeuker, 2007). The PMD sensor is capable of obtaining the distance measurements without additional mechanical or electronic devices. The sensor itself is composed by an array of pixels, which is capable of recording three layers of information:

- a brightness matrix

Figure 2.2. PMD-Technologies CamCube 3.0 is composed by a central body where the optics and sensor are located, and two LED arrays at each side of the main body for illuminating the scene.

- an amplitude matrix, which is proportional to the received signal strength
- a depth map, i.e. the distance to the objects in the scene

For this research, the depth map is used without any combination with the other two matrices. The decision for this approach is based on the harsh conditions that the camera would have as being a payload in a chaser satellite. First of all, the sun would be in the field of view (FOV) of the camera, and it may saturate the sensor. This over-exposition of light affects in many ways how the brightness intensity of a scene is captured. Secondly, the amplitude matrix is an indicator of quality in the received signal. Although this matrix could be used in environments with high-background clutter in order to detect the sources of better reflection on an object, it is expected to have only one target in the camera FOV for ADR missions.

2.1.2 ToF Camera Equipment

Based on the benefits that this kind of technology offers, this research is founded on the specifications of a PMD-Technologies CamCube 3.0 (see Figure 2.2 and Table 2.1) (PMD-Technologies, 2010). Other studies have demonstrated the pros and cons of this type of active image technology (Foix et al., 2011; Keller and Kolb, 2009), but explaining all of them here is beyond the scope of this research.

Table 2.1. PMD-Technologies CamCube 3.0 Specifications

Parameter	Value
Sensor size	200 px × 200 px
Pixel size	45µm
Field of view	40° × 40°
Focal length	12.8mm
Wavelength	870nm

2.2 Formation of Images

Although the use of brightness intensity image were not selected to be part of this research, the theory of how the images are formed in this type of sensor is relevant for the formation of the depth map. As explained in Section 2.1.1, the PMD sensor is capable of recording both intensity and distance. As matter of fact, it is valid to say that the depth map is also another type of image, where the distance to the object substitutes the brightness intensity values in the image.

2.2.1 Pinhole Camera Model

The most common model used for image formation is the perspective or pinhole model, because it mimics more accurately the behavior of real cameras. This camera model consists on the image formation on a plane by projecting the 3D points towards the center of projection O. This model assumes that the rays of light do not diverge and pass only through the center previously mentioned. Figure 2.3 illustrates the model in one dimension.

From the diagram, an object has a height H located at a distance Z from the camera. The latter has a focal length f which indicates the distance between the camera center and the image plane. By the use of similar triangles, the height of the produced image h can be obtained by the means of:

$$h = -f\frac{H}{Z} \tag{2.2}$$

The negative sign of the focal length indicates that this distance is measured in opposite direction to the location of the object, confirming also the inversion of the

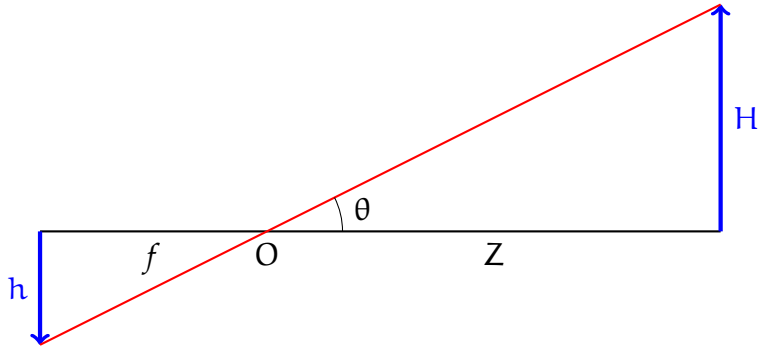


Figure 2.3. 1D Pinhole camera model. The object image is inverted in the image plane of the camera.

image height. In addition to this, the line-of-sight (LOS) angle θ is defined from the optical axis and the height of the object, as well as the height of the image, with the following relation:

$$\theta = \arctan\left(\frac{H}{Z}\right) = \arctan\left(\frac{h}{f}\right) \tag{2.3}$$

Because of the real world is 3D instead of 2D, the above formulas need to be extended, although not expressed in this research. Hence, the pinhole camera model is also extended to 3D coordinates. In order to avoid an inverted image, the camera reference frame is rotated 180° around the y_O-axis. In this manner, the image plane is located towards the object in space. The intersection of the optical axis and the image plane is called as principal point. Figure 2.4 shows the pinhole camera model with a frontal image. Now, two LOS angles are defined: a horizontal angle measured from the optical axis to the projection of the ray on the plane $x_O z_O$, and a vertical angle measured from this projection to the ray itself, as shown in Figure 2.5.

Using Equation 2.2, it could be possible to obtain the distance between a passive camera and an object if the dimensions of the latter are known in advance. However, this approach can be problematic if the object is under high dynamic behavior, i.e. translating and/or rotating. Additionally, the appropriate orientation of the object should also be known, and this is the main goal of this research. Furthermore, passive cameras suffer of scale ambiguity: taking again the 1D pinhole model for simplicity, it is possible to have different sizes of an object located at an specific distance from the center of projection. Thus, the image produced in the sensor could

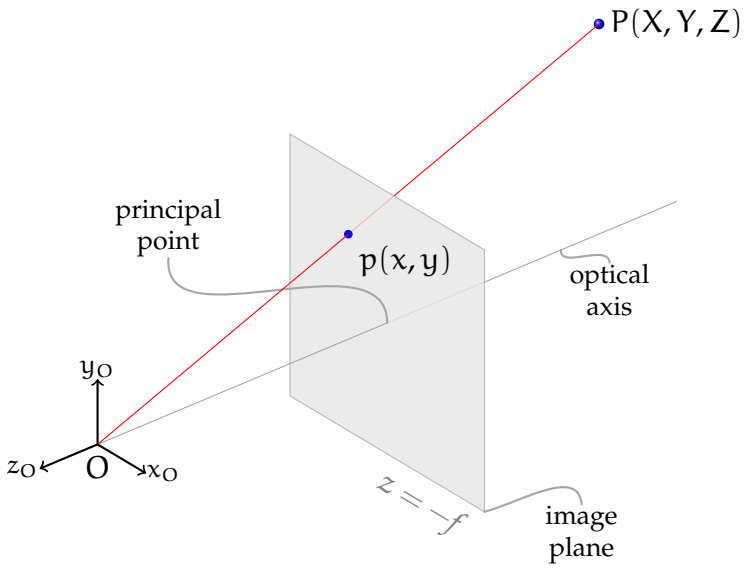


Figure 2.4. Pinhole camera model. P is a 3D point in world coordinates, and its correspondent image p, which is the intersection of the ray, which comes from P to the center of projection O, and the image plane.

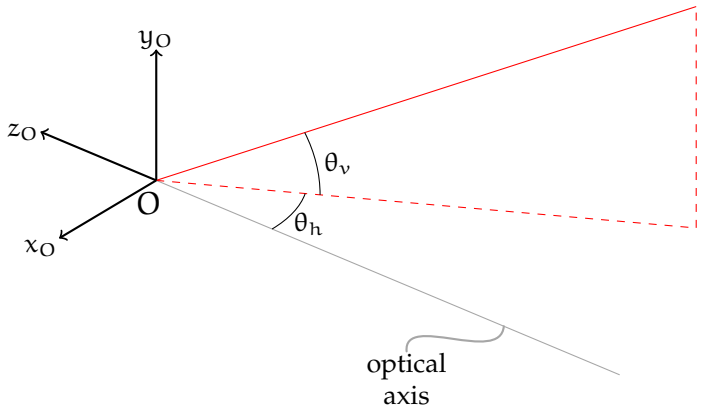


Figure 2.5. Pinhole camera model LOS angles. Horizontal angle, θ_h, and vertical angle, θ_v.

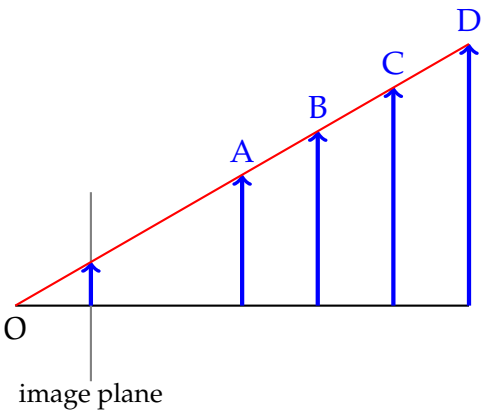


Figure 2.6. Scale ambiguity in passive cameras. Similar objects with scaled dimensions may have an equal image output.

be the same for all the objects under certain specific conditions. A new diagram showing this issue is addressed in Figure 2.6.

2.2.2 Range Imaging

In the field of computer graphics, a depth map is an image that contains information regarding to the distance between an object and a particular viewpoint. This distance information is recorded in an additional channel, usually named Z-depth or Z-buffer. The channel is also stored as an array, similar to the intensity matrix of gray-scale images, where each pixel contains information about the distance value.

This methodology has been translated into computer vision as a technique for calculating the distance to objects from a given viewpoint using cameras, which is called range imaging. Here, a depth channel is added to the 2D imaging in order to provide a distance, or range value from the specific reference frame.

Active and passive cameras are capable of obtaining a range image, based on their configuration and/or operation. For instance, one of the most common applications is the stereoscopy, where two passive cameras are properly arranged, and they simultaneously capture images for the same object. By means of matching techniques (Scharstein and Szeliski, 2002), the depth is obtained by finding corresponding points in both of the images. Perhaps the strength of this technique is also its drawback. First of all, the correspondence problem has a higher degree of

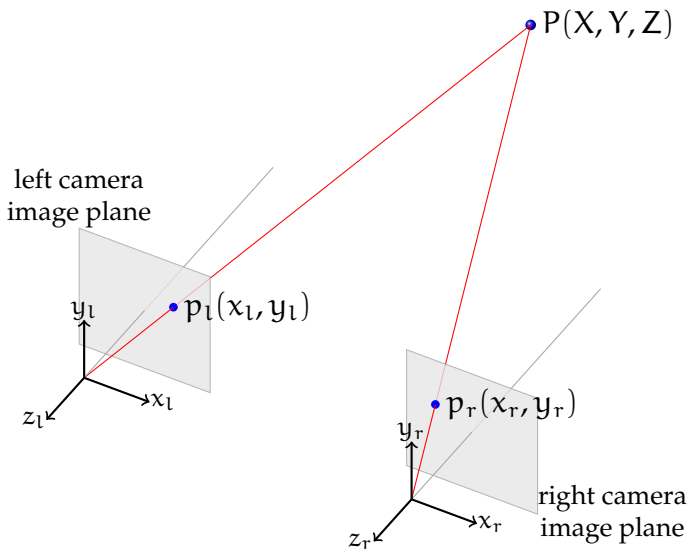


Figure 2.7. Typical stereo camera system. The light rays show a point P in global coordinates, and the respective projected scene for each camera's image.

complexity when the scene has minimum variation in intensity, color or surface texture (Rosso et al., 2013).

Additionally, the synchronization of the cameras must be ensured, i.e. the shutter time, in order to obtain the images at the same epoch. A stereo vision system can also be arranged with more than two cameras. Figure 2.7 shows a simple stereo camera setup.

Otherwise, the use of structured light also allows to obtain the range to objects using cameras (Besl, 1988). This is an active technique, because it requires an additional illumination source for its operation. The lighting system illuminates the scene with a particular light pattern (e.g. speckles, lines, dots), and this motif is observed by a camera sensor. The detected light is no longer steady but distorted because of the surface of the scene, and from those geometrical changes it is possible to reconstruct the shape of the objects, i.e. having the distance to the viewpoint. In comparison to the previous method, instead of having two cameras, one of them is replaced by the light emitter, but conserving the baseline between them. This is made in order to observe the distorted light from a different perspective than from the light source itself.

The main drawback on this technique is related to the baseline between the camera an the light emitter. Depending on the application, this baseline commands

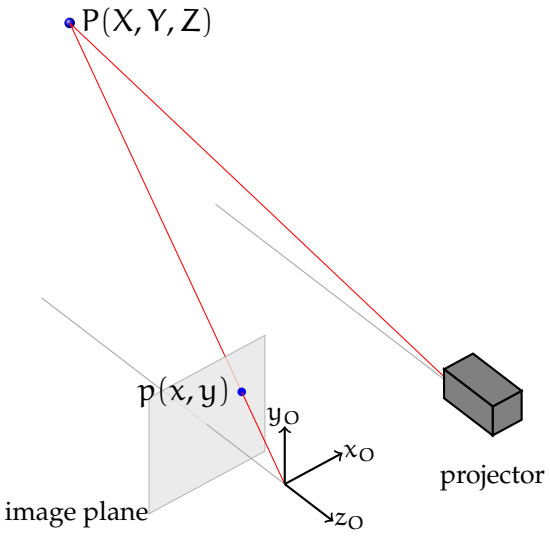


Figure 2.8. Structured light. The projector illuminates the object and the reflected light is captured by the camera. The arrangement is similar to the stereo vision system.

how far the system is capable of detecting the range. But at the same time, the power required by the illumination system is constraining the range measurement. Additionally, the computational burden of detecting the light distortion can be similar to the detection of the features in stereo imaging. A diagram of the structured light setup is shown in Figure 2.8.

Finally, lidar and ToF cameras can also obtain the called range image based on the Equation 2.1. They both have an illumination source –laser for the former and IR-LEDs for the latter–, whose reflection on the scene is captured by the sensor. Although they both have the same principle of operation, the selection of the sensor is based on different aspects, like the operative range or the environmental conditions. The outcome, however, is pretty similar for both sensors. An example of a range image can be seen in Figure 2.9.

2.3 Point Clouds

Although the range images provide the distance to the objects in query, the operations of the data supplied can be fatiguing due to the format. When the ray of a 3D point in world coordinates is projected thanks to the camera model (see Figure 2.4), the data obtained is stored as the position (u, v) in the image coordinates, plus the

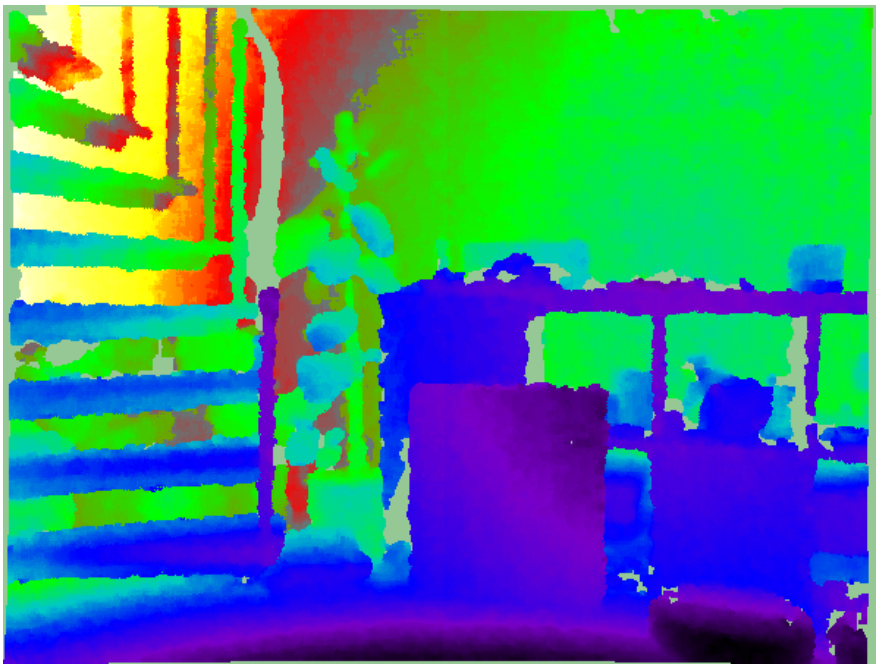


Figure 2.9. Example of a range image obtained by a ToF sensor. The color map represents the depth of the scene: cold colors are closer to the sensor.

value of the range ρ. This kind of information display can be defined as 2.5D, because the depth information in world coordinates is represented in an image of m pixels per row and n pixels per column. Most of visual navigation applications requires to have this information in full 3D, i.e. in world coordinates, because it is also required to know the constraints in each Euclidean dimension for the autonomous agent. This will ensure that a robot does not collide with the obstacles in the environment, or a manipulator can grab an object with better accuracy.

Hence, a transformation of the data is compulsory in order to have the representation of this range data as the spatial distribution of the scene under observation. With the a priori knowledge of the internal characteristics of the ToF camera, like the number of pixels of the images and size of the pixel, as well as the focal length of the camera, the conversion of data can be performed using Algorithm 2.1.

See that, for each range value ρ, a point in 3D space P_{ij} has coordinates (X, Y, Z) in the same units as the range, usually meters. Because of the data transformation was performed in the pinhole model reference frame, the coordinates of each of the obtained points are also given in this canonical frame. The new set that contains all new points is now considered as a point cloud. This term refers to the similitude

Algorithm 2.1 Convert range image into point cloud

```
1: procedure RANGE2POINTCLOUD(m, n, f, ρ)
2:     S ← ∅
3:     for i ← 0 to m do
4:         for j ← 0 to n do
5:             v ← (i, j, f)
6:             v̂ ← v / ‖v‖
7:             P_ij ← ρv̂
8:             S ← S ∪ P_ij                ▷ append P_ij to a new list
9:         end for
10:    end for
11:    return S
12: end procedure
```

between the data spatial distribution, and a common cloud in the sky. A sample of a point cloud was shown in Figure 1.5. Thanks to this data transformation, it is possible to distinguish the surface of the objects (unless some of them are overlapped), as well as to get rid of the scale ambiguity, inherent to the pinhole model of the passive cameras.

The point clouds are now full 3D representations of the objects, i.e. it is possible to measure the height, width, and depth, although the users are seeing the data on a screen. Depending on the capture device, e.g. stereo camera or Kinect™ (Soares Beleboni, 2014), the points can also have additional labels associated, like the color of the object's surface from where the ray comes, the reflected signal amplitude in case of ToF cameras, or the distance to each point similar to that on range images.

From this representation, the point cloud expands the possibilities on how we can visualize the information, like the dispersed dots in space with or without the color information. Alternatively, the range can be shown in a color scale, where the cold colors indicates the objects closer to the viewpoint, opposite to the warm colors (see Figure 2.9). If the concept of volume is somehow missing, the possibility of showing boxes that enclose the points based on given conditions, like the length of each box or the number of points per box. Those boxes are called "voxels" (Kaufman et al., 1993), and they were named as an analogy for the 2D pixels, but in 3D. Apart of giving an idea of volume, this voxel representation is useful in tasks like point query search, because the voxels can be considered like buckets and each of them can be labeled depending on the analysis requirements. An example of the "voxelization"

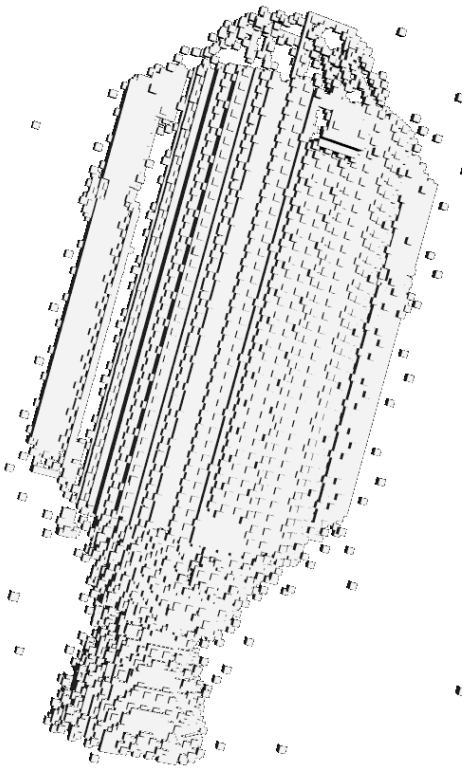


Figure 2.10. Visualization of voxels based on the rocket body point cloud.

of the point cloud obtained from the Cosmos-3M CAD model (see Figure 1.4) can be seen in Figure 2.10.

Additionally, meshes can be formed based on the points' position, giving a more conceptual representation of the surface of the objects in the scene. The latter gives the idea of how the objects are in reality, but this representation has not enough influence in signal processing. Furthermore, the meshes demand more computational power when the scene need to be updated due to camera manipulation, i.e. panning or zooming.

2.4 Summary

This chapter presented an overview of image theory, related to the procurement and further representation of 3D information using passive and active cameras. First of all, the explanation of the operation principle of the ToF cameras, including a short description of a capturing device, was given in order to establish the concepts for

understanding this range measurement technique. Moreover, the depiction of how the mathematical model of a pinhole camera serves for obtaining a range image, and the consequent point cloud. Finally, diverse representations of the point clouds were briefly described, starting with the basic form like the distribution of points in 3D space, and finishing with the mesh representation, which can be useful for visual inspection and object collision.

3 Point Clouds: Features, Segmentation and Registration

As described in Chapter 2, point clouds are a manner of spatial representation of scenes, observed by either an active camera or a stereo vision arrangement. The spatial data is now a sparse distribution in 3D. Therefore, it is required to operate and manipulate the data in order to extract quantifiable information, additional to their Euclidean position, which allows a better understanding of the scene nature.

Based on this reason, the fact of having additional clues about the distribution of the points in Euclidean space can allow to perform tasks that are trivial for humans, but extremely hard for computers, e.g. to recognize an object, to estimate the position of an object, to figure out how fast an object is moving, or to grab an element without the risk of breaking it or dropping it.

This chapter presents some clarifications about the basic operation on point clouds, as the estimation of the surface's normal vectors. This procedure is the main foundation for further applications like point feature descriptors, also briefly explained in the current section. Additionally, the method for segmenting point sets using geometric primitives, as well as the registration of multiple clouds, are also subject of explanation in this chapter.

3.1 Surface's Normals Estimation

The point cloud obtained by an active camera is just the position of scattered points with respect to a reference frame, i.e. the sensor, that represents the surface of the objects in the viewed scene. But the sensor does not have the capability of distinguishing how the orientation of the observed surfaces is. For this reason, the

estimation of the surface's normals is one of the most important operations in the manipulation of point clouds. The expected outcome of this process is a vector field, i.e. the normals, whose orientation in space will help to identify and differentiate the different objects in a scene, or the object's faces that it may have. As a reminder, the normal of a surface is the perpendicular vector to the tangent plane associated to that surface.

This process required a number of steps in order to obtain a well-descriptive normal vector field. Thus, the pipeline for this estimation is composed by the following steps:

- Subdivision of the points in the cloud to create small patches.
- Estimation of the surface's normal vectors.
- Validation of the vector field orientation.

3.1.1 Surface Division

It is compulsory to detect the surfaces that belong to the different objects in the scene. Sometimes the point clouds do not define surfaces by their own, then a division of the point cloud is required in order to describe the clusters, i.e. small surface patches, to estimate their normals. But the selection of those patches is crucial not only for the accuracy of the estimation, but also because of the data distribution of the point clouds to be manipulated. One possible method to understand the clusters is comparing them to the neighborhood concept. The latter is defined as the area around a specific place, e.g. a query point in the cloud, and the additional elements that are located in their vicinity, i.e. the points around the query point. Therefore, the vicinity of the points are determined by two different methods:

- All the points enclosed in an imaginary sphere of certain radius, whose center is the query point.
- The closest number of points of the query point, no matter their distance to that point.

The selection of one of those methods to define the surface clusters heavily depends on how the points are distributed. Additionally, the proper selection of

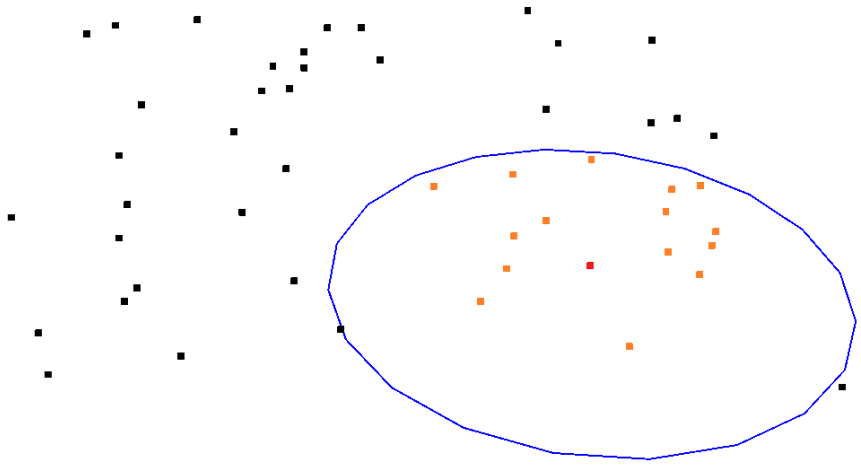


Figure 3.1. Definition of surface cluster based on distance. All the points inside the imaginary circle of a given radius from a query point (in red) are defined as neighbors (in orange). The latter conform the surface patch, which will be then used to estimate the normal of the plane tangent to the surface at the query point.

either the radius for the first method, or the number of neighbors for the second one, is a limitation when an automated process of the data manipulation is required. An example of the first method, radius neighborhood, is shown in Figure 3.1. As a rule of thumb, the selection of those parameters –radius or number of neighbors– is made by trial and error, depending on the level of detail required for each specific application.

3.1.2 Normal Vector Field

With the determination of the surface patches, the geometrical information of a local region can be estimated by means of the normal of the patch. The latter is a vector in 3D, which has an specific orientation to a given reference frame. This outcome is important not only for knowing the orientation of a surface, but also for identifying objects, if they are known a priori.

Different normal estimation methods have been evaluated (Klasing et al., 2009), and one of the simplest available methods is to determine the normal to the plane tangent to the requested patch at the query point. Having a patch $\mathcal{P}$ composed by P_i points, being k the number of points in the patch, the query point $\mathbf{X}$ can be defined

as follow:

$$\mathbf{X} = \bar{\mathrm{P}} = \frac{1}{k}\sum_{\mathrm{i}=1}^{k} \mathrm{P}_{\mathrm{i}} \tag{3.1}$$

This query point is also considered the centroid, $\bar{\mathrm{P}}$, of $\mathcal{P}$. With this into account, the creation of a covariance matrix $\mathcal{C}$ of the subset $\mathcal{P}$ is:

$$\mathcal{C} = \frac{1}{k}\sum_{\mathrm{i}=1}^{k} (\mathrm{P}_{\mathrm{i}} - \bar{\mathrm{P}})(\mathrm{P}_{\mathrm{i}} - \bar{\mathrm{P}})^{\mathsf{T}} \tag{3.2}$$

Now, it is possible to find the eigenvalues and eigenvectors of this new base matrix, i.e.

$$(\mathcal{C} - \lambda_{\mathrm{j}}\mathbf{I}_3)\vec{v}_{\mathrm{j}} = 0 \text{ , with } \mathrm{j} \in \{1, 2, 3\} \tag{3.3}$$

The eigenvectors $\vec{v}_{\mathrm{j}}$ are the principal components of the set $\mathcal{P}$, and their correspondent eigenvalues are λ_{j}. This procedure is called Principal Component Analysis (PCA) (Pearson, 1901). The intention of obtaining those elements –eigenvectors and eigenvalues– is to fit the points in the patch to an ellipsoid, where each of the three axes represent a principal component. At the same time, PCA can be considered as a coordinate frame transformation where the eigenvectors are the new orthogonal axes of the new reference frame located at the query point $\bar{\mathrm{P}}$. Each principal component shows the degree of correlation based on the direction of the eigenvector and its associated eigenvalue. The bigger the latter is, the greater the correlation of the data in that direction. In mathematical terms, having $0 \leqslant \lambda_1 \leqslant \lambda_2 \leqslant \lambda_3$, the eigenvector $\vec{v}_3$ encloses the direction of maximum correlation. Relating those PCA concepts to the estimation of the normal of a tangent plane, the latter is defined by the vectors associated to the biggest and middle eigenvalues. Otherwise, the eigenvector related to the lowest associated eigenvalue is the approximate normal vector to the surface under analysis:

$$\lambda_1 \rightarrow \vec{v}_1 = \pm\vec{\mathrm{n}} = \begin{bmatrix} \mathrm{n}_x \\ \mathrm{n}_y \\ \mathrm{n}_z \end{bmatrix} \tag{3.4}$$

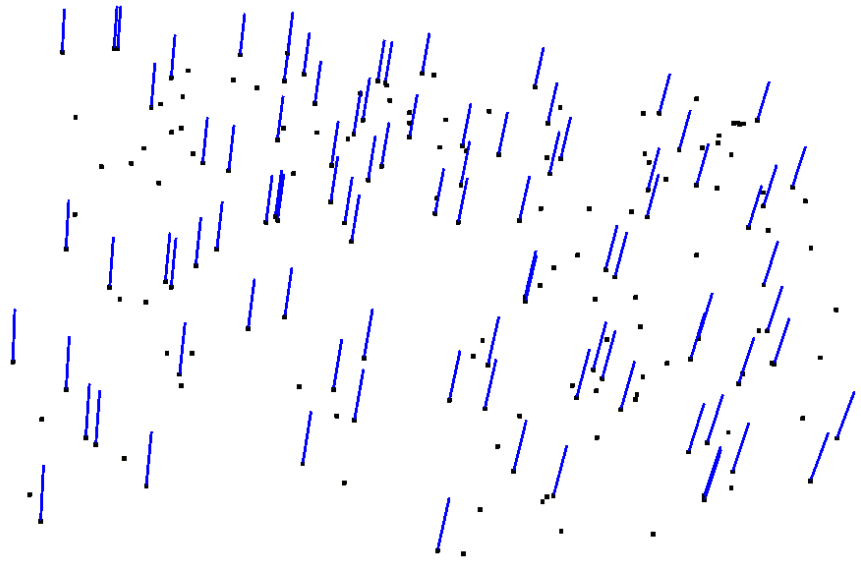


Figure 3.2. Example of a surface's normals estimation in a point cloud, shown in blue.

3.1.3 Validation of Normals' Orientation

Because PCA is not capable of resolving the sign of the eigenvectors, the result can be ambiguous in the normals' orientation. This is the reason of the $\pm$ sign in Equation 3.4. An additional step is required in order to ensure the consistency in the normals' orientation. As the location of the sensor reference frame is known, and the point clouds are acquired for this coordinate frame, the orientation of the normals $\vec{n}$ must point towards the sensor. In other words, this operation ensures that the camera is seeing the external surface of the objects, which is facing towards the sensor. This is figured out by means of:

$$\vec{n} \cdot (O - P_i) > 0 \tag{3.5}$$

Sometimes the viewpoint is not known, depending on the application. For instance, when more than one sensor are operating at the same time, but the point clouds are simultaneously concatenated in a new coordinate system. For this thesis, the use of a single ToF camera always provides the same viewpoint location for all the point clouds acquired.

3.2 Point Feature Descriptors

The surface's normal is a basic geometrical descriptor, which helps to identify the geometry relations and constraints of point clouds. But it could happen that many objects in the scene may have similar normal orientation, reducing the information about any specific surface. If the normal fields of different objects in a scene share comparable characteristics, this may lead to erroneous analysis for further development due to the emergence of false positives in the information.

To reduce the impact of bad accuracy, the point feature descriptors have been developed in order to provide better geometrical information of a subset of the point cloud. Principally, a descriptor is a means used for comparing point clouds, and it has robustness dealing with rigid transformations, i.e. rotations and/or translations among the point sets. The descriptors should be also effective against the noise that the data could have. Finally, the descriptors must be similar, if not identical, when the point cloud resolution changes. Hence, the final purpose of the point feature descriptors is to identify a point across many different point clouds, as long as the data is referring to the same scene/object. Either the camera or the objects may have rotations and translations in space, and the descriptors must be identifiable no matter those transformations.

Each descriptor –also called signature– can be described as a histogram, whose bins encapsulate different values, which are based on the metric employed to build the signature, e.g. the distance between points in a selected neighborhood, or the normal orientation angle for a determined surface. It is important to bear in mind that the number of bins may or not affect the point cloud processing velocity. As the final purpose is to identify a point in two or more clouds, the comparison of its descriptor may take more or less time, depending on the type of signature and the length of the histogram. For instance, for real time applications it should be better to have shorter descriptors. If the time is not a concern, longer histograms can be more accurate at the matching procedure.

The signatures can be classified in two big categories: local and global. First, the local descriptors express how the local geometry surrounding the query point is, without previous knowledge of the object as a whole. Table 3.1 shows a summary of the feature local descriptors, including the number of histogram bins for each of them. Otherwise, the global descriptors require the whole object in order to enclose

Table 3.1. Point Feature Local Descriptors

Name	Number of Bins
SHOT - Signatures of Histograms of Orientations	352
PFH - Point Feature Histogram	125
FPFH - Fast Point Feature Histogram	33
3DSC - 3D Shape Context	1980
USC - Unique Shape Context	1960
RSD - Radius-based Surface Descriptor	289
Spin images	153

Table 3.2. Point Feature Global Descriptors

Name	Number of Bins
VFH - Viewpoint Feature Histogram	308
CVFH - Clustered Viewpoint Feature Histogram	308
OUR-CVFH - Oriented, Unique and Repeatable CVFH	308
ESF - Ensemble of Shape Functions	640
GRSD - Global Radius-Based Surface Descriptor	20

its geometry. Thanks to this, it can be possible to identify the objects, estimate the position and orientation, and extract the shape. Table 3.2 presents a survey of global descriptors, also indicating the number of bins in the histogram. Additional and detailed information about the both local and global descriptors is available (Aldoma et al., 2012a), but only a small selection of them are related to the development of this research work, and they will be shortly described in Sections 3.2.1 and 3.2.2. In the end, the selection and usage of the point signatures are based on the expected outcome and the processing speed required for a specific application.

3.2.1 Local descriptors

The use of local descriptors has been useful in passive imaging, when they are employed to detect similar features in stereo camera setups (Bonin-Font et al., 2008). The idea behind this is to export the capability from 2D images to 3D point clouds. Below is explained in more detail two of the more relevant local feature descriptors.

3.2.1.1 Point Feature Histogram

The Point Feature Histogram (PFH) is one of the most common local descriptors in the point cloud processing algorithms (Rusu, 2010). The PFH encloses the geometrical information around a query point based on the normal direction of the point's neighborhood. If the normal estimation is not constant and pretty accurate, the quality of this descriptor is directly affected and the outcome's precision is really poor.

The formulation of the histogram is composed by the following steps: first, it is required to have the normal estimation of the point cloud under evaluation. Then, the query point and the normal associated to its surface are the base for the histogram formation. Taking into account the points in the vicinity, the algorithm pairs each point in the region not only with the query point, but also with the other points in the neighborhood (see Figure 3.3).

Afterwards, for each couple of points in the list, a pair of new fixed coordinate frames is computed based on the points' associated normals. A surface reference frame is created at the query point, using the following definition:

$$\begin{aligned} u &= n_q \\ v &= u \times \frac{(P_s - P_q)}{\|P_s - P_q\|} \\ w &= u \times v \end{aligned} \tag{3.6}$$

Having established the main reference frame, a copy of it is translated to each of the neighbors of the query point. Therefore, three angular variables can be calculated as follows:

$$\begin{aligned} \phi &= \arccos\left(u^T \cdot \frac{(P_s - P_q)}{\|P_s - P_q\|}\right) \\ \alpha &= \arccos\left(v^T \cdot n_s\right) \\ \theta &= \arctan2\left(w^T \cdot n_s, u^T \cdot n_s\right) \end{aligned} \tag{3.7}$$

Figure 3.4 shows an example of the surface reference frame, and the angles used for the PFH descriptor formation. When this procedure is performed for all the point couples, the resultant values are binned to created a histogram for each angular

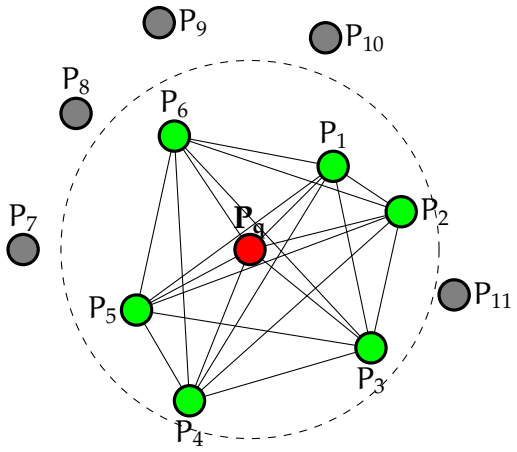


Figure 3.3. Vicinity region for PFH calculation. The query point (red) and its neighbors are connected among themselves.

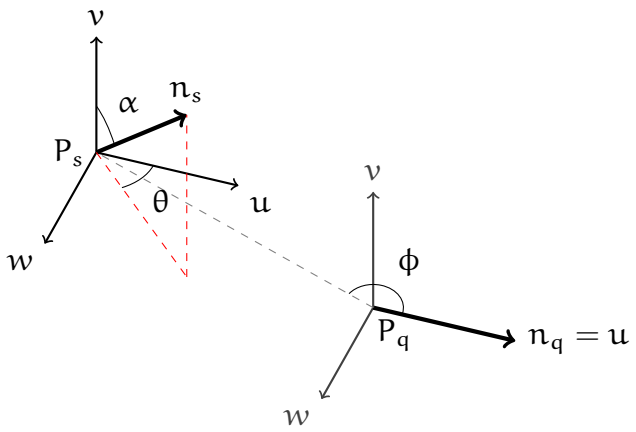


Figure 3.4. Coordinate frames for angular and distance feature calculation in PFH.

value and one for the euclidean distance between each couple. An example of this descriptor can be seen in Figure 3.5.

3.2.1.2 Fast Point Feature Histogram

Although PFH is a very stable descriptor with great accuracy, it also has the disadvantage of being a huge computational burden in order to perform the descriptor calculation in real time. For that reason, a simplification method was proposed in order to reformulate the signature without affecting its descriptiveness. This was called Fast Point Feature Histogram (FPFH) (Rusu et al., 2009), which reduces the

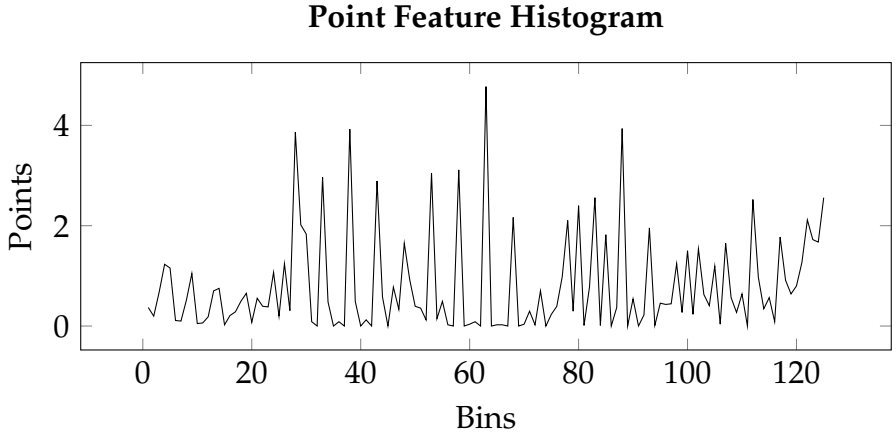


Figure 3.5. Example of Point Feature Histogram descriptor for a single query point.

computational complexity from $\mathcal{O}\{o \cdot k^2\}$ for the former descriptor to $\mathcal{O}\{o \cdot k\}$ for the latter, being k the number of points in the neighborhood.

Because of this new descriptor is a modification of the previous one, it maintains the same principle of connecting the query point to its direct neighbors and calculate the angular features for each couple using Equations 3.6 and 3.7. Afterwards, a new vicinity is defined for each of the points linked to the query one (see Figure 3.6), and the process is repeated again. In the end, the FPFH is the integration of all the values previously calculated, without including the distance between points as a differentiating factor. However, it was indeed included as a weighting coefficient, because the angular values are counted twice: from the query point to a neighbor and from the latter to the former, when the new vicinity is created. Figure 3.7 shows the descriptor based in this new calculation method.

3.2.2 Global Descriptors

As mentioned in Section 3.2, the global descriptors can be useful for object pose estimation. Because of this, these signatures can be considered as the first candidates for solving the space debris pose estimation. Hence, a brief explanation of three of them is necessary in order to understand how these descriptors can be employed to solve the position and orientation determination problem.

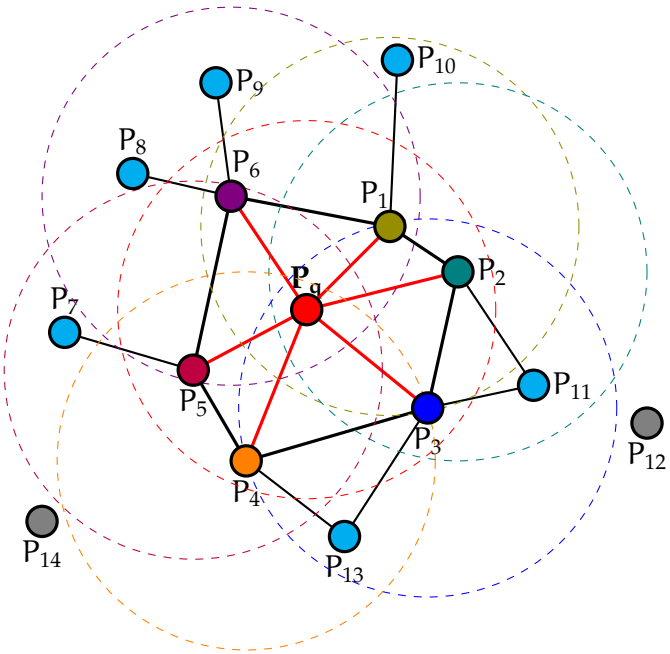


Figure 3.6. Vicinity region for FPFH calculation. The query point is linked to its neighbors. Similarly, each of the neighbors are connected to their own neighbors.

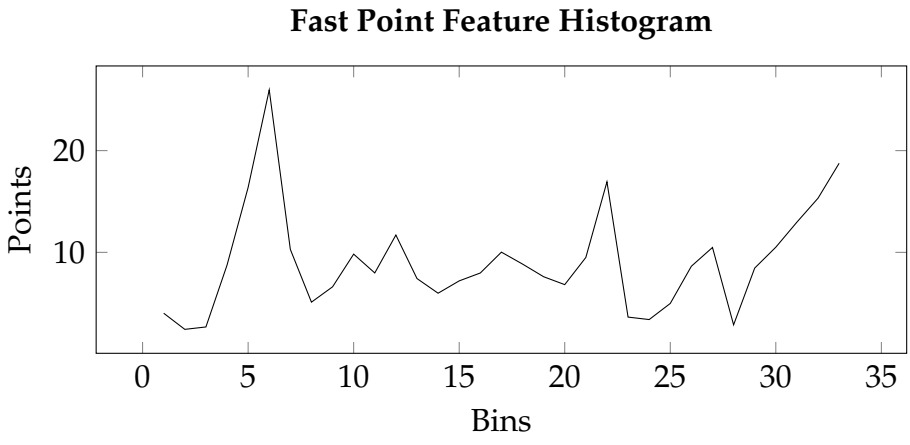


Figure 3.7. Example of Fast Point Feature Histogram descriptor for a single query point. Notice the reduced number of bins (33) in comparison to the PFH (125).

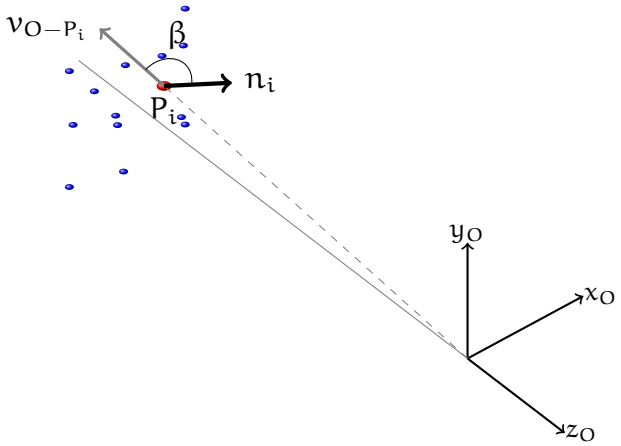


Figure 3.8. Calculation of the viewpoint angle with respect to a point cloud at a query point (red).

3.2.2.1 Viewpoint Feature Histogram

This is the application of the FPFH with the inclusion of the viewpoint information, but using the whole point cloud instead of a subset of it (Rusu et al., 2010). This signature was named Viewpoint Feature Histogram (VFH), thanks to this new metric inclusion. In this particular case, the query point is just the centroid of the point cloud –assuming that only one object is observed– using the Equation 3.1. With the obtained centroid, the angular values for the histogram formation are calculated for each point in the cloud paired with the centroid. In addition the viewpoint, i.e. the sensor origin, is used to create a new metric to each point whose normal has been estimated previously using the next expression:

$$\beta_i = \arccos\left(n_i \cdot \frac{O - P_i}{\|O - P_i\|}\right) \tag{3.8}$$

being the subscript i each point and its correspondent normal. Figure 3.8 shows how is the formation of this new acquired angle. The histogram concatenation proceeds as usual as for the FPFH, and an example can be seen in Figure 3.9.

3.2.2.2 Clustered Viewpoint Feature Histogram

Because of the stability weakness shown by VFH regarding sensor noise/errors and occlusions, the Clustered Viewpoint Feature Histogram (CVFH) tries to solve those drawbacks. The purpose is to subdivide the surface into subsets, or clusters, where

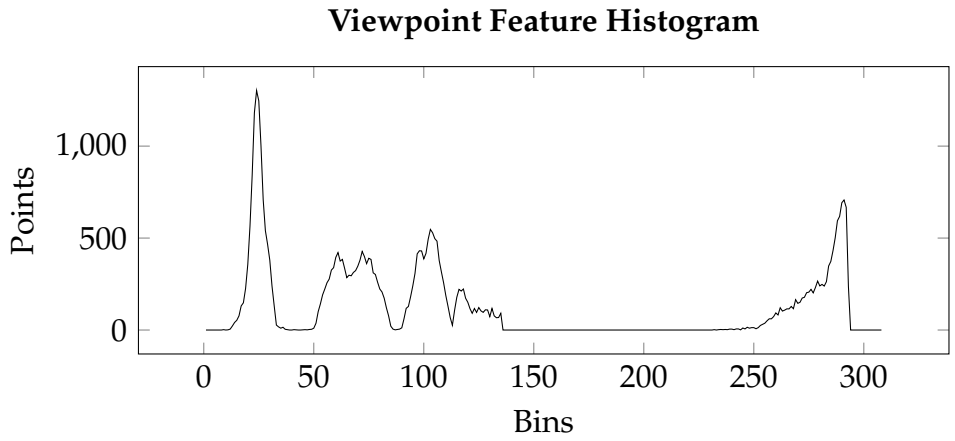


Figure 3.9. Example of Viewpoint Feature Histogram descriptor for the whole rocket body. The bins from $0-45$, $46-90$ and $91-135$ encapsulates the angles information. The viewpoint component is enclosed in last 63 bins.

a VFH can be computed for each region (Aldoma et al., 2011). The conditions for generating the point cloud clusters are established as follows:

$$\begin{aligned} \|P_i - P_j\| &< t_d \\ n_i \cdot n_j &> t_n \end{aligned} \tag{3.9}$$

The latter establishes the two thresholds, distance and normal direction, that define which points belong to the same cluster. This condition permits to exclude the spurious points, whose surface normal orientation is wrong estimated most of the times. An example of a clustered point cloud can be seen in Figure 3.10.

Additionally to the segmentation of the cloud, a Shape Distribution Component (SDC) was added to the VFH descriptor giving an additional robustness factor. SDC is defined as:

$$SDC = \frac{\left(\bar{P} - P_i\right)^2}{\max\left(\left(\bar{P} - P_i\right)^2\right)} \tag{3.10}$$

with $\bar{P}$ as the centroid of each cluster, using Equation 3.1. This new descriptor component encodes the points distribution around the cluster's centroid. For instance, this allows to distinguish two planar surfaces one from another, although having similar normal orientations.

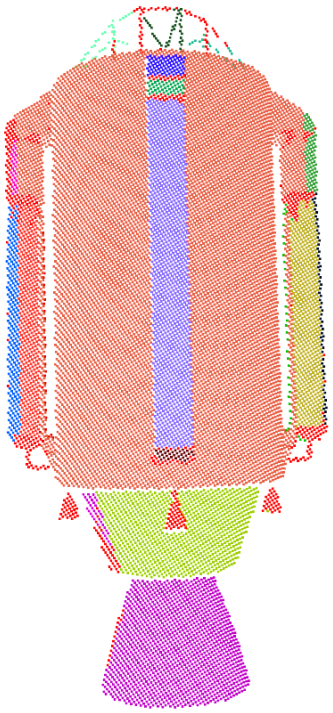


Figure 3.10. Segmentation of a point cloud in clusters, based on the normal orientation and distance among elements.

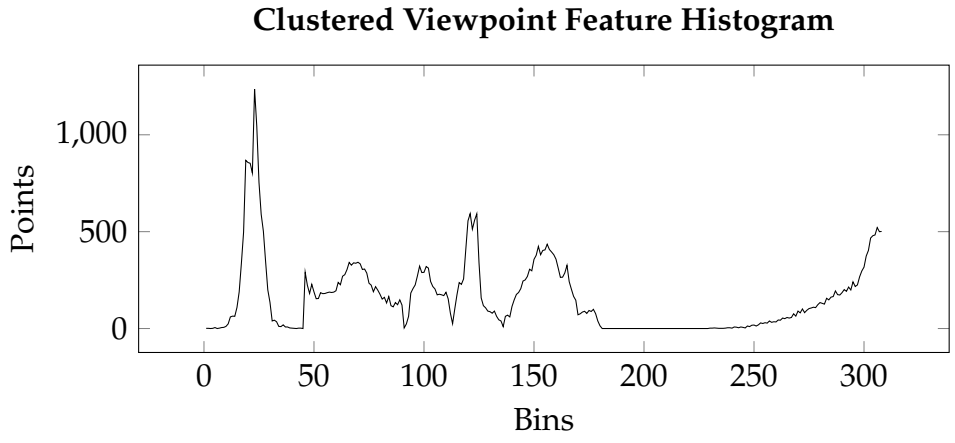


Figure 3.11. Example of Clustered Viewpoint Feature Histogram descriptor, using the same rocket body pose as used for calculating VFH shown in Figure 3.9. The SDC is encapsulated in bins $146 - 190$.

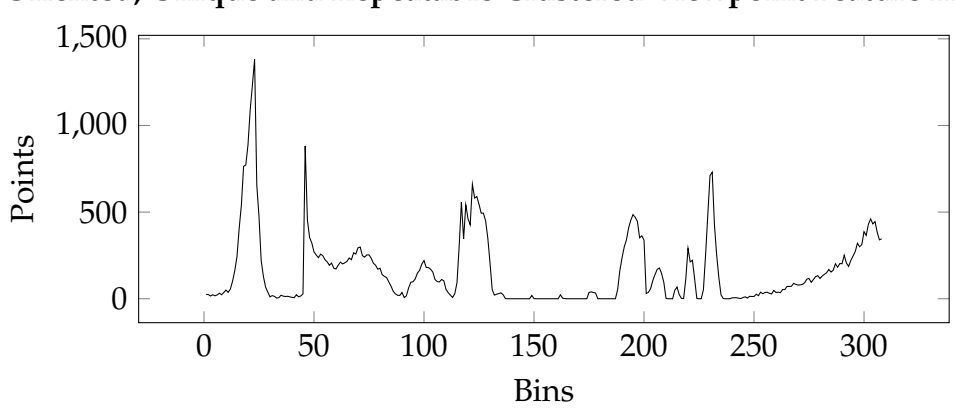


Figure 3.12. Example of Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram descriptor, using the same rocket body pose as used for calculating VFH shown in Figure 3.9. The SGURF is encapsulated in bins $146 - 245$.

3.2.2.3 Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram

The Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram (OUR-CVFH) has included a new metric in order to enhance the performance of the previous descriptor. OUR-CVFH has the same operation principle of the CVFH, with the difference of having a second filtering for each of the clusters obtained. Thus, it has a new thresholding condition after the outcome of the Equation 3.9, where the angle between the average normal of the cluster and each point in the latter is in certain range. As the new regions are better defined than before, the generation of a new surface coordinate frame can be easily computed (Aldoma et al., 2012b), called as Semi-Global Unique Reference Frame (SGURF).

Because of the method enhancement, the orientation of each of the SGURFs in the cloud can be discriminated, obtaining the same coordinate frame orientation no matters the position of the object. Additionally, this new frame in the refined cluster allows to calculate a new feature based on the points location for each octant of the SGURF, defined by the sign of the axes: $(x^+, y^+, z^+) \dots (x^+, y^-, z^-) \dots (x^-, y^-, z^-)$. This new component replaces the SDC from the CVFH signature, as can be seen in Figure 3.12.

3.3 Random Sample Consensus

Random Sample Consensus (RANSAC) is a method used to fit a mathematical model to a sample of given data with a number of iterations previously established (Fischler and Bolles, 1981). The RANSAC algorithm divides the data into inliers and outliers, being the former those data that have an acceptable margin to fit into the model, and the latter those elements that cannot fit at all. RANSAC is a non-deterministic algorithm, i.e. a procedure that, even with the exact input for a given number of trials, the result can be different for each of those runs. For the case of the RANSAC, the result has certain probability and it improves with every new iteration, although without reaching 100%. The procedure is shown as follows:

Algorithm 3.1 Random Sample Consensus

Require: p_s, p_f, M, N, $\vec{x}$, K_t
1: $L \leftarrow \frac{\log(p_f)}{\log(1-(p_s)^M)}$
2: **for** $i \leftarrow 0$ **to** L **do**
3: Select M random items from data
4: Estimates $\vec{x}$ on N
5: $K \leftarrow$ number of elements from N that fits on $\vec{x}$
6: **if** $K \geqslant K_t$ **then**
7: **return** K set fits the model based on $\vec{x}$
8: **end if**
9: **end for**
10: **return** No data items fit the model

From Algorithm 3.1, p_s is the probability of selecting data randomly that may belong to the requested model, p_f is the probability that the algorithm will finish without finding a good model fit (even if it exists), M is the minimum number of data that can allow the estimation of the model parameters, N is the total number of elements in the data set, $\vec{x}$ is a vector that contains the model's parameters, and K_t is the threshold for determining if the data can fit the model or not. For instance, the simplest example of fitting a model in 2D sparse data is using a line model and see how many of the items can fit into the model. In the case of point cloud processing, the RANSAC algorithm may procure if the point set can be fitted to 3D primitives like planes, cones, spheres, cylinders, among others.

3.3.1 Plane Model

The plane is one of the models most used in RANSAC to be applied on point clouds, specially in terms of indoor environment perception, where it is necessary to detect planar surfaces like the floor, the walls, and the ceilings. The detection of those structures can allow the robot to map them. Also a lot of human-made objects are composed by planes like a table, a door, or a closet. In case of outdoor navigation, the urban roads, highways, and building's facade can also be considered as planes. In case of roads, however, there are depressions and steep streets that may distort the smoothness of it. For OOS and ADR, most of satellites have planar surfaces and displayed solar arrays that can be fitted to a plane model.

First of all, the general equation of the plane is defined as:

$$ax + by + cz + d = 0 \tag{3.11}$$

and its normal vector, $\vec{n}$, is

$$\vec{n} = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{3.12}$$

Having a point P, that belongs to the plane, the Equation 3.11 can be rewritten in algebraic form:

$$\vec{n} \cdot P = -C \tag{3.13}$$

for some constant C. Equation 3.13 is the Hessian normal form of the plane. Under the assumption that the normal vector has magnitude 1, it is possible to define that

$$d = -\vec{n} \cdot P \tag{3.14}$$

The minimum number of points required to calculate the plane equation is three. This is valid because three non-linear points define a triangle, which is the minimum plane possible in Euclidean geometry. From those points, two vectors are created in order to calculate the cross product between them. The outcome is the plane's normal vector candidate (see Equation 3.12). And using Equation 3.14, the independent coefficient d of the plane's general equation is calculated.

In addition to this, it is necessary to calculate the perpendicular distance, r, to any 3D point, Q, to a plane, by means of:

$$r = \vec{n} \cdot Q + d \tag{3.15}$$

The sign of r indicates in which side of the plane is located the point: positive if it is on the side where the normal vector points, or negative otherwise. Although it is important, the sign is not relevant for the plane fitting. However, a certain distance threshold, r_t, must be defined in advance, using the relation:

$$0 \leqslant r_{P_i} \leqslant r_t \tag{3.16}$$

Therefore, the steps for detecting planes is shown in Algorithm 3.2. This procedure is repeated L iterations until the number of inliers support the found plane model (see Algorithm 3.1). An example of an identified plane in a point cloud can be seen in Figure 3.13.

Algorithm 3.2 Plane model fitting

Require: $\mathcal{P}$, r_t
1: $\mathcal{S} \leftarrow \varnothing$
2: Select three non-collinear random points $\{P_1, P_2, P_3\} \in \mathcal{P}$
3: Compute $\vec{n}$, and d ▷ Eq. 3.11 - 3.14
4: **for** $i \leftarrow 0$ **to** N **do**
5: $\quad r_{P_i} \leftarrow \vec{n} \cdot P_i + d$
6: $\quad$ **if** $0 \leqslant r_{P_i} \leqslant r_t$ **then** ▷ Eq. 3.16
7: $\quad\quad \mathcal{S} \leftarrow \mathcal{S} \cup P_i$
8: $\quad$ **end if**
9: **end for**
10: **return** $\mathcal{S}$

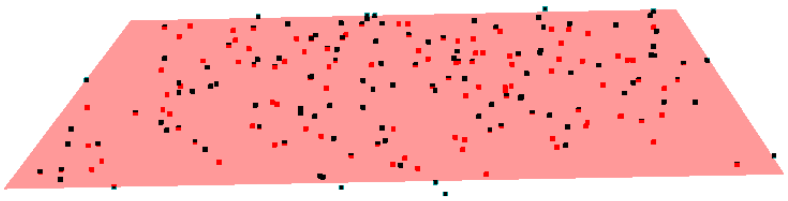


Figure 3.13. Plane model fitted to a point cloud.

3.3.2 Cylinder Model

The second model to be considered is the elliptical cylinder, due to the fact that most of rocket launchers has a body with a cylindrical shape, and it is defined as:

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 = 1 \tag{3.17}$$

where a and b are the semi-major and semi-minor axes of the cross-section ellipse, respectively. In the case those values are equal, i.e. $a = b$, Equation 3.17 becomes valid for a circular cylinder, hereafter referred to as a cylinder.

The initial step for the validation of the cylinder model is the previous normal estimation for the whole point cloud, using the procedure explained in Section 3.1. Then, the iterative process of the RANSAC method takes part (see Algorithm 3.1), when two points and its associated surface's normal are selected randomly from the point set. Afterwards, the cross product between those two normal vectors is performed. The resultant vector of this operation has the same direction akin the cylinder's axis of symmetry, as long as the normals chosen belongs to the cylinder's surface. In the case the cross product is very close to zero, i.e. the normals selected were almost parallel, the points are discarded and the selection of a new pair is performed. In mathematical terms, the direction of the cylinder's axis is given by:

$$\vec{a} = \vec{n}_1 \times \vec{n}_2 \tag{3.18}$$

Under the assumption that both normal vectors are valid, a parametric line is created using for each normal and its associated point, by means of

$$\mathcal{L} = P + s\vec{n} \tag{3.19}$$

where P is either one of the point randomly selected and $\vec{n}$ is the associated normal unit vector. Equation 3.19 is known as the parametric equation of a line in 3D, where any line can be expressed by a point that lies on the line, the unit vector that indicates the direction of the line, and the parameter $s \in \mathbb{R}$ that suggests how far from the point the line is extended. If $s > 0$, the line extends in the same direction of $\vec{n}$. Otherwise, the line extends to the opposite direction of $\vec{n}$ if $s < 0$.

Returning to the cylinder model fitting, a local coordinate frame is located in such a way that its x-axis is aligned with one of the normal vectors, i.e. $\vec{n} \cdot x = 1$ and z-axis is aligned with the cylinder's axis $\vec{a}$, i.e. $\vec{a} \cdot x = 0$. Then, both parametric lines from the two normal vectors are projected into the local xy plane, and their intersection is calculated equating $\mathcal{L}_1$ and $\mathcal{L}_2$. This point is then the center C of the cross-section circle, whose distance r_c to the surface's points should be equal to the radius r of the cylinder model. This verification can be done using a variation of the Equation 3.17, as follows:

$$x^2 + y^2 = r_c^2 \tag{3.20}$$

with a certain limit threshold given by:

$$r_{min} \leqslant r_c \leqslant r_{max} \tag{3.21}$$

where the minimum and maximum values of r are previously settled. This procedure is repeated L times (see Algorithm 3.1) until there is enough confidence in the number of points that are fitted into the cylinder model.

After the model fitting, a new "linear" point cloud is created with all the "intersection points" previously found, whose average point also lies on the cylinder's axis. Moreover, a new set of the axis vectors estimated for each point couple is generated, with their direction to be consistent. The cylinder's axis is the average of all the vectors in the set. Finally, the average of all the radius values obtained is calculated to define the last parameter of the model. In summary, a point that belongs to the axis, the axis' direction unit vector, and the radius are the outcome parameters for the model. Algorithm 3.3 compiles the procedure previously described. Figure 3.14 shows the cylinder model fitting, including the axis of symmetry.

Algorithm 3.3 Cylinder model fitting

Require: $\mathcal{P}$, $\mathcal{N}$, r_{max}, r_{min}

$\mathcal{M} \leftarrow \varnothing$
$\mathcal{A} \leftarrow \varnothing$
$\mathcal{R} \leftarrow \varnothing$
while $\mathcal{P}$ is not $\varnothing$ **do**
  Select two random points $\{P_1, P_2\} \in \mathcal{P}$ with $\{\vec{n}_1, \vec{n}_2\} \in \mathcal{N}$
  $\vec{a} \leftarrow \vec{n}_1 \times \vec{n}_2$ ▷ Eq. 3.18
  $\mathcal{L}_1 \leftarrow P_1 + s_c\vec{n}_1$, $\mathcal{L}_2 \leftarrow P_2 + t_c\vec{n}_2$ ▷ Eq. 3.19
  Calculate C, with $\mathcal{L}_1 = \mathcal{L}_2$ and solving for s_c and t_c
  $r_{c,1,2} \leftarrow \sqrt{x_{1,2}^2 + y_{1,2}^2}$ ▷ Eq. 3.20
  if $r_{min} \leqslant r_{c,1,2} \leqslant r_{max}$ **then** ▷ Eq. 3.21
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{P_1, P_2\}$
    $\mathcal{A} \leftarrow \mathcal{A} \cup \vec{a}$
    $\mathcal{R} \leftarrow \mathcal{R} \cup r_c$
  end if
  $\mathcal{P} \leftarrow \mathcal{P} \setminus \{P_1, P_2\}$
end while
$P_{mean} \leftarrow \frac{1}{k}\sum_{i=1}^{k}(P_i \in \mathcal{M})$
$\vec{a}_{mean} \leftarrow \frac{1}{k}\sum_{i=1}^{k}(\vec{a}_i \in \mathcal{A})$
$r_{mean} \leftarrow \frac{1}{k}\sum_{i=1}^{k}(r_{c,i} \in \mathcal{R})$
return P_{mean}, $\vec{a}_{mean}$, r_{mean}

3.4 Iterative Closest Point

Iterative Closest Point (ICP) (Besl and McKay, 1992) is a method to register point clouds, when one of them has been transformed in terms of rotation and translation. This procedure of alignment is called as registration, where the main purpose is to match the points of one cloud into the other. The latter is only valid when both point clouds are equal. At the end, the outcome of the ICP algorithm is composed by the rotation matrix and the translation vector required to overlap one cloud into the another. The method is iterative until a criterion of convergence is achieved, which is usually the minimum distance, d_{max}, between the point clouds.

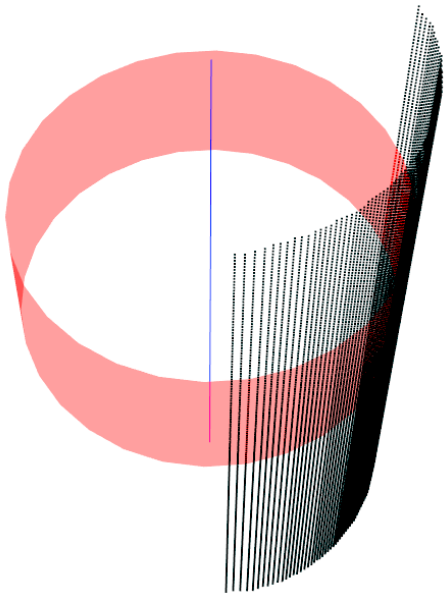


Figure 3.14. Cylinder model fitted to a point cloud. The blue line refers to the cylinder's longitudinal axis.

For the ICP, one point cloud is kept as reference, or *target*. Otherwise, the second point set, called as the *source*, is transformed to match the target as best as possible. Hence, having two point sets, $P_i \in \mathcal{P}$ as the source, and $Q_i \in \mathcal{Q}$ as the target, for $i = \{1, 2, \ldots, N\}$,

$$Q_i = \mathbf{R}P_i + \mathbf{t} + N_i \tag{3.22}$$

where $\mathbf{R}$ is the rotation matrix, $\mathbf{t}$ is the translation vector, and N is a noise vector. The purpose of ICP is to minimize the Euclidean distance between the two point clouds (Arun et al., 1987), by means of:

$$\arg\min f(\mathbf{R}, \mathbf{t}) = \frac{1}{N}\sum_{i=1}^{N} \|Q_i - (\mathbf{R}P_i + \mathbf{t})\|^2 \tag{3.23}$$

ICP employs linear algebra to determine the transformation, specifically the use of Singular Value Decomposition (SVD), whose routine is condensed in Algorithm 3.4. Having the two point clouds, a covariance matrix (see Equation 3.2) is constructed using the centroid of each point cloud (see Equation 3.1). From the new

matrix, the SVD is applied in order to find three matrices:

$$\mathcal{C} = USV^T \tag{3.24}$$

being U and V as the matrices that contain the left and right singular vectors of the covariance matrix, respectively, and S as the non-zero singular values matrix. The rotation matrix between the two clouds is the multiplication between V and the transpose of U:

$$\mathbf{R} = VU^T \tag{3.25}$$

thus, obtaining the translation vector:

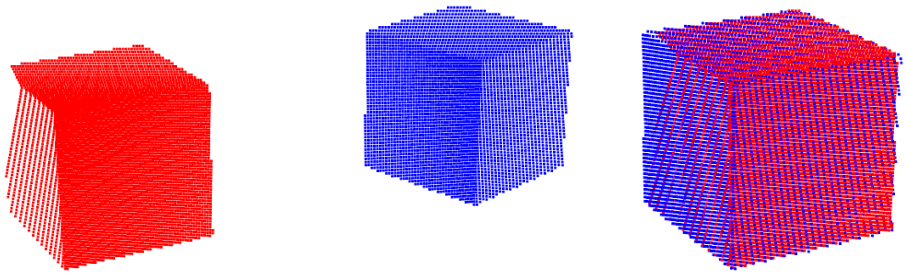
$$\mathbf{t} = \bar{Q} - \mathbf{R}\bar{P} \tag{3.26}$$

Algorithm 3.4 Singular Value Decomposition

Require: $\mathcal{P}$, $\mathcal{Q}$

1: $\bar{P} \leftarrow \frac{1}{N}\sum_{i=1}^{N} P_i$ ▷ Eq. 3.1
2: $\bar{Q} \leftarrow \frac{1}{N}\sum_{i=1}^{N} Q_i$ ▷ Eq. 3.1
3: $\mathcal{C} \leftarrow \sum_{i=1}^{N} \left(P_i - \bar{P}\right)\left(Q_i - \bar{Q}\right)^T$ ▷ Eq. 3.2 variant
4: $\mathcal{C} \leftarrow USV^T$ ▷ Eq. 3.24
5: $\mathbf{R} \leftarrow VU^T$ ▷ Eq. 3.25
6: $\mathbf{t} \leftarrow \bar{Q} - \mathbf{R}\bar{P}$ ▷ Eq. 3.26
7: **return** $\mathbf{R}$, $\mathbf{t}$

ICP is widely used for creating maps in robot navigation (May et al., 2008), because it allows to know the environment's position and orientation with respect to the sensor. On the other hand, if the ToF camera is placed fixed and the objects are moving in its FOV, then is possible to obtain the pose of further point clouds if the position and orientation of the object is known in a certain epoch. For instance, the latter refers to the situation of a chaser satellite and its target, in order to identify the object's pose. Algorithm 3.5 shows the process for obtaining the rigid transformation between two point clouds. Based on the point clouds shown as sample in Figure 1.5, the ICP procedure was executed in order to register both point clouds: Figure 3.15a shows two point clouds before the use of the ICP method. Figure 3.15b exposes the outcome of the ICP algorithm for the registration process.



(a) Source (blue) and target (red) point clouds before ICP.

(b) Point clouds alignment after ICP

Figure 3.15. Point cloud registration using Iterative Closest Point algorithm.

Algorithm 3.5 Iterative Closest Point

Require: $\mathcal{P}$, $\mathcal{Q}$, d_{max}

1: $\mathbf{R}$, $\mathbf{t}$ using SVD ▷ Algorithm 3.4
2: **while** not converged **do**
3: **for** $i \leftarrow 0$ **to** N **do**
4: $Q_i \leftarrow \texttt{ClosestPointQ}(\mathbf{R}P_i + \mathbf{t})$
5: **if** $\|Q_i - (\mathbf{R}P_i + \mathbf{t})\| \leqslant d_{max}$ **then**
6: $w_i \leftarrow 1$
7: **else**
8: $w_i \leftarrow 0$
9: **end if**
10: **end for**
11: $\mathbf{R}, \mathbf{t} \leftarrow \arg\min \sum_{i=1}^{N} w_i \|\mathbf{R}P_i + \mathbf{t} - Q_i\|^2$ ▷ Eq. 3.23 variant
12: **end while**
13: **return** $\mathbf{R}$, $\mathbf{t}$

3.5 Summary

This chapter exposed the fundamental operation on point clouds as is the surface's normal vectors estimation, being the basic feature descriptor for a 3D point set. Additionally, the point feature signatures have been described, using as foundation the normal vectors previously explained. Here, two main categories, local and global descriptors, were depicted, showing their capabilities, strengths and drawbacks.

Furthermore, the Random Sample Consensus (RANSAC) was determinant for fitting geometric models into the data, and detecting them from the point distribution. Finally, the Iterative Closest Point (ICP) algorithm was presented as one of the most reliable methods for registering successive point clouds, in order to either reconstruct the object or determine its pose under certain conditions.

4 | Pose Estimation

This chapter presents the fundamental problem to be treated, as is the pose estimation of a non-cooperative target. Resembling Chapter 1, the reckoning of the position and orientation of an object in OOS is a cooperative task, because the target has markers previously designed for giving aid to the chaser satellite. Principally, the purpose of the fiducials is to assist the system in giving a first pose estimate based on the location of the target's visual markers. This is not possible with non-cooperative targets, either defunct satellites, or rocket bodies, because they were not designed for further servicing at the design phase. The main intention was to left them "floating" in orbit after EOL.

For that reason, it is necessary to employ new methods to recover the space debris before they can be considered a full-potential threat. Generally, the target geometry is known a priori, in order to develop the mission to capture a certain family of targets. Thus, model-based pose estimation procedures (Horaud et al., 1989) are methods for estimating the position and orientation of the target. Some studies (D'Amico et al., 2012; Kanani et al., 2012) have shown different procedures by using passive cameras, i.e. detecting the target, find the edges in the image and match the obtained wireframe model with the designated model previously stored in the chaser satellite's system. Nevertheless, this approach is subject to prone error due to illumination variations.

Because of the latter, the pose estimation using 3D point clouds will have an enhancement related to the 2D edges methods. First of all, the point clouds provide the 3D coordinates of the target surface. Secondly, the illumination depends only on the ToF camera, under certain orbit conditions.

4.1 Estimation Problem Description

The autonomous rendezvous begins when there is visual relative navigation between the chaser spacecraft and the target, i.e. the passive camera has received light reflected from the target. This can be classified into the far range phase of the rendezvous maneuver, previously explained in Section 1.2. The distance between the chaser and target could be up to tens of kilometers, depending on the optical head of the camera. The next phase is the close range, which is comprised between tens of meters up to 10km. Usually, a combination of passive cameras and lidar are commonly proposed for OOS missions. The final approach phase is the step just previous to the physical encounter of the two objects on orbit. The operation range is between 1 to 3m and approximately 20m. Here, the chaser spacecraft must have great certainty in the position and orientation parameters estimation, in order to perform the capture maneuver in a safe way. Basically, a portion of the final approach phase is overlapped with the close range one, and there is no clear boundary between them. Hence, the phases are designated depending on the type of target to be captured: a big object can be detected from a farther distance than a smaller debris.

Due to the potential operation of a ToF camera in ADR, the scenario for estimating the position and orientation of a target is assumed in this way:

- The chaser spacecraft is located at a safe distance from the target, performing station keeping at the same nominal altitude of the latter.
- There are neither external forces nor torques acting on the target.
- Both chaser spacecraft and target are located in quasi-circular orbit.
- The target cannot control its attitude, supposed to be tumbling/rotating.
- The target is non-cooperative, i.e. it has no visual markers, and it does not provide information about its state.
- The target must have a distinctive geometry, in order to determine the 6DOF.

A little disclaimer about the last item: due to the fact that the pose estimation is solely based on the target's geometry, there should exist a section of the structure that

helps to define the body reference frame without ambiguity. For instance, a cubesat or a rocket body, which are commonly symmetrical about the longitudinal axis, could only provide 4DOF –3 for position and only 1 for orientation–. Nevertheless, sometimes the absolute orientation is not required, depending on the method selected for docking/capture (Shan et al., 2016). For instance, if the methods employed are either a net or a harpoon, there would not be need of a full pose estimation. In the contrary, if the capture means proposed is a robotic arm –also assumed for this thesis–, the latter will grab the rocket body from a protruding section, like the nozzle or a flange. Hence, this action requires a full pose estimation to ensure the grasp, and to avoid a possible impact between the agent and the debris.

The state vector of the debris to be estimated by the chaser satellite is composed by the position of the target $\mathbf{r}$, its derivative, i.e. the linear velocity $\dot{\mathbf{r}}$, the attitude orientation $\boldsymbol{\alpha}$, and the angular velocity $\boldsymbol{\omega}$. This can be written in a 1-column vector as:

$$\mathbf{x} = \begin{bmatrix} \mathbf{r} \\ \dot{\mathbf{r}} \\ \boldsymbol{\alpha} \\ \boldsymbol{\omega} \end{bmatrix} \tag{4.1}$$

with each element defined as follows:

$$\mathbf{r} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \qquad \dot{\mathbf{r}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}, \qquad \boldsymbol{\alpha} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}, \qquad \boldsymbol{\omega} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{4.2}$$

The attitude orientation can be described by means of the Euler angles: ϕ is the roll angle, i.e. the rotations around the x-axis of the body coordinates frame; θ is the pitch angle, which encloses the rotations around the y-axis, and ψ is the yaw angle, which defines the heading when the body rotates around the z-axis.

Although the Euler angles are easily understandable, they can cause problems due to gimbal lock and rotation ambiguity –different angles can refer to the same rotation–. Because of the previous reasons, the target's attitude will be described using quaternions, because they have no ambiguity in describing a rotation, and they are also easier to handle by computers, making them more efficient than the Euler angles. Hence, the attitude parameter $\boldsymbol{\alpha}$ of the state vector $\mathbf{x}$ is now changed

to the quaternion $\mathbf{q}$, and is defined as:

$$\mathbf{q} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{4.3}$$

where q_s is the scalar part of the quaternion, and $\mathbf{q}_v$ is the quaternion's vectorial part:

$$\mathbf{q} = \begin{bmatrix} q_s \\ \mathbf{q}_v \end{bmatrix}; \qquad q_s = \begin{bmatrix} q_0 \end{bmatrix}, \qquad \mathbf{q}_v = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{4.4}$$

4.2 Initialization

The final goal is to estimate the position and orientation of a rocket body with respect to a chaser satellite. The rocket's upper stage is a non-cooperative target, which is assumed to be tumbling on orbit without control. When the chaser has acquired a precise estimation of the target's dynamics, it would be possible to perform a safe capture, reducing the probabilities of having a collision, which can produce additional debris.

In order to achieve great accuracy at the pose estimation, a settlement for the dynamic parameters of the target is of vital importance. This is called as *initialization*, which is defined as the a priori knowledge that an agent has about an object's position and orientation in 3D space.

The initialization step refers to the procedure to obtain the initial values of the target's state vector $\mathbf{x}$ with respect to the sensor reference frame, in which all the measurements are performed. This means that a fixed coordinate frame has to be defined previously in the target's body in order to compute the changes in translation and rotation with respect to the camera. In the case of the rocket body, the fixed reference frame is located at the cylinder's centroid, with the z-axis aligned with the cylinder's axis of symmetry. The positive region of the axis points towards the instrumentation compartment, i.e. opposite to the nozzle.

Nevertheless, the location of the x- and y-axis can have infinite possible locations in the transverse section of the body. But for the specific case of the Cosmos-3M second stage (see Figure 1.4), it is possible to eliminate the ambiguity location of the xy plane. Due to the external configuration, two outer cylindrical tanks can help to point one of the body reference frame axis and define a fixed location for the coordinate frame. However, a third point for determination is still required. Fortunately, a fairing was built just 90° apart from the lateral tanks, giving the possibility of excluding any uncertainty on the definition of the coordinate frames. Recapping, the location and orientation of the body fixed reference frame is as follows:

- The origin of the coordinates frame is located at the cylinder's centroid.
- The z-axis is aligned with the cylinder's axis of symmetry.
- The x-axis points towards the second stage's "frontal" fairing.
- The y-axis completes the triad, pointing to the axis of symmetry of one of the lateral tanks, in such a way that $y = z \times x$.

Figure 4.1 shows a better depiction of the target body frame. The Euler angles were described in Section 4.1 for a configuration where the x-axis is aligned with the longitudinal axis of the vehicle. However, the definition of the rocket body's reference frame differs from that explanation. Here, the roll angle ϕ is measured around the body's z-axis; the yaw angle ψ is measured around the x-axis; and the pitch angle θ is measured around the y-axis. This arbitrary definition of the body reference frame was decided in order to match the axes of the 3D CAD model.

Thus, the visual navigation system must estimate the parameters of the target's state vector based on the declaration of the body reference frame. To achieve this objective, two methods were assessed in order to accomplish this goal: by means of point clouds' global descriptors, and through recognition of point clusters associated to distinctive target's external geometry.

4.2.1 Initialization Based on Global Descriptors

Under the assumption that the whole rocket body is observed by the ToF camera and no other additional object, it is possible to recognize the pose of the target based on

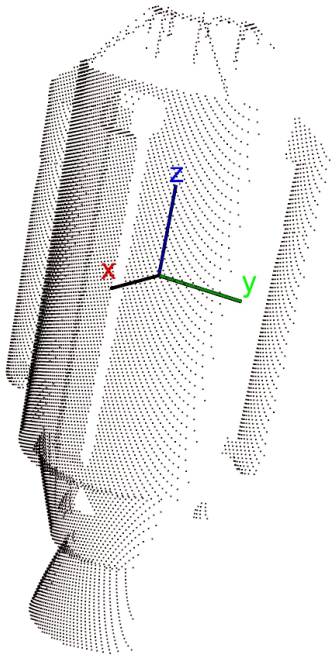


Figure 4.1. Definition of the rocket body fixed frame. It is located at the centroid of the main cylinder body; z-axis is aligned with the cylinder's axis of symmetry and pointing towards the bay for the navigation instruments; x-axis points towards the fairing; y-axis points towards a lateral tank.

global descriptors, defined in Section 3.2.2. This procedure is composed by two main steps: the first one is to produce a training database filled with global signatures of the object at different poses, and a second phase where a test point cloud descriptor is matched against all the elements stored in the training data (Gomez and Boge, 2015).

4.2.1.1 Training phase

In this initial step, it is required to create a database, where the elements are global descriptors of the designated target. The particularity of those signatures is that each of them is computed from a different pose of the object of interest. The main idea is to populate the database with different histograms associated with its correspondent pose. The method for obtaining this training data set is by taking many depth images of the object at different pre-established poses. For each of them, the point cloud is obtained, and the chosen global descriptor is calculated for each of the point sets previously recovered.

For every day objects, it would not be so difficult to find the appropriate element to take the point clouds at different poses. Otherwise, if the object is not so common, or the target is too big, the construction of the data base may be more difficult. For that reason, another alternative to produce the training database is using 3D CAD models of the object, and produce the point clouds by simulating the sensor. Due to the fact that this thesis uses as a target an object longer than 6m, the last option is preferred for the procedure assessment. Based on that, the CAD model shown in Figure 1.4 was the object from which synthetic point clouds were obtained using software tools, specifically BlenSor (Gschwandtner et al., 2011). The latter is an open-source add-on, that not only simulates ToF cameras, but also scanning lidars and light-structured range sensors, like the Kinect™. This add-on runs on top of Blender[1], an open-source 3D creator software.

Hence, it is possible to obtain any number of views as wished for the object poses. Then, the global descriptor is computed for each of those poses. When this calculation is performed, it is possible to build the training database, which is composed by:

- The point cloud of a specific pose view.
- The global descriptor associated to its correspondent point cloud view.
- The respective object's position and orientation to the point cloud view.

The last item usually is declared as the rigid transformation from the body fixed reference frame to the sensor coordinates frame, which is previously known because it was assigned on purpose.

4.2.1.2 Testing phase

When the training list is finally completed, then it is feasible to estimate any given pose of the target. After obtaining the actual point cloud view of the target at an unknown pose, the correspondent global descriptor is calculated. Now the algorithm would try to find the closest match between the current view descriptor and all those previously stored in the training database. This descriptors comparison is

[1] `www.blender.org`

performed by brute force, based on the Chi-square distance between histograms (Pele and Werman, 2010), whose computation is performed by means of:

$$\chi^2 = \frac{1}{2}\sum_{i=1}^{N}\frac{(H_{i,\mathcal{P}} - H_{i,\mathcal{Q}})^2}{H_{i,\mathcal{P}} + H_{i,\mathcal{Q}}} \tag{4.5}$$

where N is the number of bins in the histograms. Hence, this metric performs a bin-to-bin comparison. The ideal match is when the χ^2 distance is equal to 0, where the current view's descriptor has been already trained previously. For that reason, the outcome of the histogram comparison will be the list with the best possible candidates, i.e. those whose distance is close to 0, that could be represented by the current view descriptor.

As a final step, it is required to calculate the rigid transformation between the current point cloud view and the found nearest neighbor by means of the ICP algorithm (see Section 3.4). As a requirement, all the point clouds stored in the training database are then considered as the target cloud for the ICP procedure, and the current point cloud view is defined as the source cloud. Hence, ICP tries to align the current view to the found match.

However, the transformation from the target cloud to the source one is required, because the demanded pose is that which refers to the current view, i.e. the source point cloud. For instance, any orientation of the rocket body is described by the orthonormal matrix $\mathbf{R}$, which rotates the sensor reference frame $\mathcal{F}$ into the body fixed frame $\mathcal{F}_b$:

$$\mathcal{F}_b = \mathbf{R}\mathcal{F} \tag{4.6}$$

On the other hand, the rotation matrix from the ICP, $\mathbf{R}_{ICP}$, indicates the rotation from the current view pose, $\mathcal{F}'_b$, to the nearest match, $\mathcal{F}_b$. Also, although the translation $\mathbf{t}_{ICP}$ that defines the displacement between source and target point clouds, it is not taken into account for estimating the orientation. Hence, both poses can be expressed in the same way as Equation 4.6:

$$\mathcal{F}_b = \mathbf{R}_{ICP}\mathcal{F}'_b \tag{4.7}$$

The required pose is the current view one, though. With this in mind, the pose of the test view is obtained from modifying Equation 4.7:

$$\mathcal{F}'_b = \mathbf{R}^T_{ICP}\mathcal{F}_b \tag{4.8}$$

Finally, combining Equations 4.6 and 4.8, the current pose of the rocket body with respect to the sensor coordinates frame is:

$$\mathcal{F}'_b = \mathbf{R}^T_{ICP}\mathbf{R}\mathcal{F} \tag{4.9}$$

4.2.2 Initialization Based on Body Geometry

The initialization method explained in Section 4.2.1 is based on nearest neighbor search among a training list populated by descriptors, obtained by a real object or a CAD model of the target. When we refer to space debris, the shape of the target is unknown, and the structure might be affected by external factors like impacts with other debris or metal degradation. If this happens, the probabilities of finding a closer match in the database could be drastically reduced, and there would not be any possibility of extracting the target's initial pose.

Accordingly, it could also be possible to obtain the position and orientation of the debris without any additional information, but the general outer geometry of the target. Most of human-built objects are composed by geometrical primitives, i.e. planes, circles, triangles, et cetera, for 2D; and cubes, cylinders, cones, spheres for 3D, among others. Taking the advantage of this, and with the assumption of knowing some geometrical information a priori about the rocket body:

- The main body is a circular cylinder, with known radius.
- The number of appendices that the body has, i.e. external tanks, fairings, etc.
- The position of the rocket nozzle on one of the body's extremities.

4.2.2.1 Recognition of the main body

Thanks to the procedure of the recognition of primitives by the use of the RANSAC algorithm (see Section 3.3.2), the detection of the rocket body as the target for ADR

can be achieved using the cylinder model. When the fitting process is successful, the outcome is the estimation of the cylinder longitudinal axis, i.e. the parametric equation of the line in 3D space, as well as the points set that conform the main body or the rocket's upper stage.

4.2.2.2 Estimation of the cylinder's centroid

Generally speaking, the point P found in the axis estimation routine does not coincide with the middle point of the cylinder. As it can be reminded, origin of the body reference frame is located at the cylinder's centroid (see Section 4.2). Hence, the detection on this point in the estimated cylinder's axis of symmetry is crucial for a good pose initialization.

Taking the points set that fitted onto the cylinder model, each point Q_i is projected onto the axis of symmetry as:

$$t_i = \vec{d}^{\,\intercal} \left(Q_i - P \right) \tag{4.10}$$

obtaining the parameter t for each of those points, in such a way that the new coordinates of the projected point Q_i' are:

$$Q_i' = P + t_i \vec{d} \tag{4.11}$$

which is the same equation of a parametric line (see Equation 3.19). This new "linear" point cloud is located along the cylinder's axis, whose extremal values of t_i determine the two longitudinal ends of the cylindrical body, Q'_{min} and Q'_{max}. The centroid M is then the resultant average of those two end points. Figure 4.2 shows the centroid of the cylinder, as well as the longitudinal end points of the cylinder's axis of symmetry.

4.2.2.3 Identification of the nozzle, external tanks and fairing

Although the RANSAC algorithm for detecting the cylinder provides the direction of the line that lies on the rocket's axis, the estimation of the unit vector direction cannot be previously settled (Gomez Martinez and Eissfeller, 2016), e.g. to point towards the camera, like the procedure made for the point cloud's normals estimation. Due

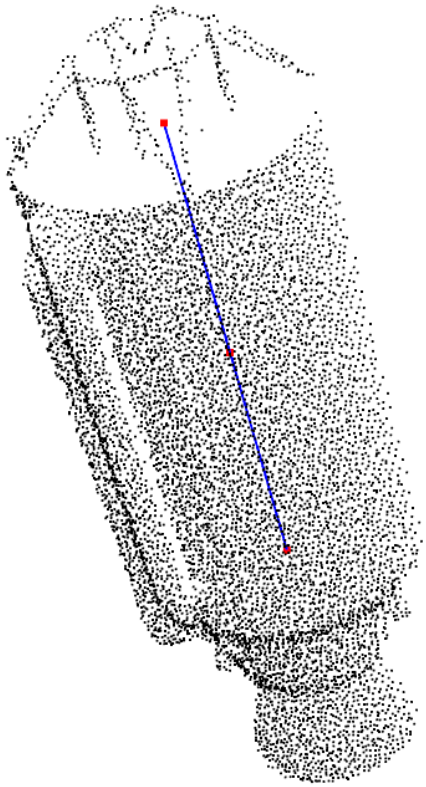


Figure 4.2. Estimation of cylindrical axis and rocket body's centroid. The centroid is the average of the axis extremal points Q'_{min} and Q'_{max}.

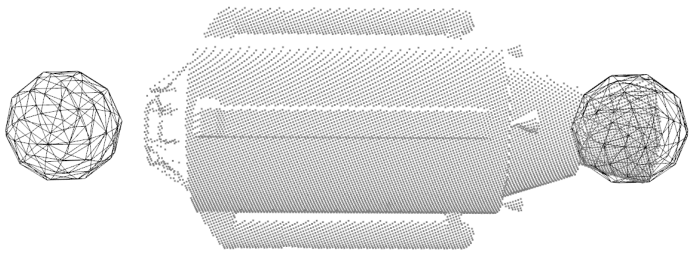


Figure 4.3. Visualization of the nozzle detector at both possible locations along the cylinder's axis of symmetry.

to the definition of the body fixed reference frame (see Section 4.2), it is required to have this vector $\vec{d}$ to be pointing towards the instrumentation bay of the upper stage, i.e. in opposite direction of the rocket's nozzle.

For achieving this goal, a cluster detector isolates a spherical region, whose center is located at a given distance along the axis from the main cylinder's centroid in the direction of $\vec{d}$. The radius of this sphere is selected so that the number of points lying within the filter is much larger than the number of points encapsulated when the detector is located in the direction of the instrumentation bay, as shown in Figure 4.3.

Additionally, the normal vector of the point cluster inside the spherical filter is again estimated. In such a way, it is possible to increase the certainty about the nozzle, because their surface's normals tend to intersect the cylinder's axis of symmetry, which should be the same for the rocket nozzle of the Cosmos-3M upper stage.

After the detection of the two main structures of the rocket body, i.e. main cylindrical body and nozzle, the points associated to those surfaces can be eliminated, letting the additional appendages more accessible for further calculations. The following step to carry on is the detection of one of the lateral tanks. Because they are cylinders with known radius as the main body, the detection of those structures followed the same procedure detailed in Section 3.3.2. The detection of one of the two lateral tanks is enough for the pose initialization.

Finally, the last structure to be identified is the fairing, located on the surface of the main cylindrical body. A new cluster detector is applied with a known distance from the cylinder's centroid, with the correspondent sphere's radius. Thanks to the upper stage's footage before launch, this fairing is a protrusion composed by planar surfaces. Hence, it is possible to validate the point cluster as part of the fairing if

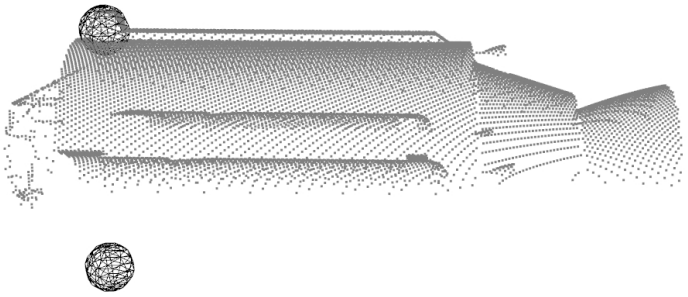


Figure 4.4. Visualization of the fairing detector at both possible locations.

planar surfaces are detected by using RANSAC for planes fitting, as explained in Section 3.3.1. Similarly to the nozzle spherical filter, Figure 4.4 shows the possible locations that the fairing detector can have. The flowchart shown in Figure 4.5 summarizes the sequential steps performed to extract the necessary geometrical information from the available point cloud.

4.2.2.4 Definition of the body pose

The z-axis of the body reference frame is defined first, using as its principal vector $\mathbf{k}$ the direction of the cylinder's axis $\vec{d}$. If the nozzle is located in the opposite direction of $\vec{d}$, then $\mathbf{k}$ is correct for the coordinate frame definition. Otherwise, the vector $\vec{d}$ should be flipped, i.e. changing the sign.

The principal vector $\mathbf{j}$ of the y-axis is determined by the unit vector from the cylinder's centroid towards the axis of the detected lateral tank, while $\mathbf{i}$ completes the right-hand frame by pointing towards the side of the main cylinder where the fairing lies (see Figure 4.1).

Afterwards, the orientation of the rocket body is then described via the orthonormal matrix $\mathbf{R}$, formed by the retrieved vectors in such a way that:

$$\mathbf{R} = \begin{bmatrix} \mathbf{i}_x & \mathbf{i}_y & \mathbf{i}_z \\ \mathbf{j}_x & \mathbf{j}_y & \mathbf{j}_z \\ \mathbf{k}_x & \mathbf{k}_y & \mathbf{k}_z \end{bmatrix} \tag{4.12}$$

At last but not least, the translation of the target is defined by the vector $\mathbf{t}$, which is formed by the distance between the camera coordinates frame $\mathcal{F}$ and the main cylinder's centroid M. Therefore, the pose initialization for the rocket body is

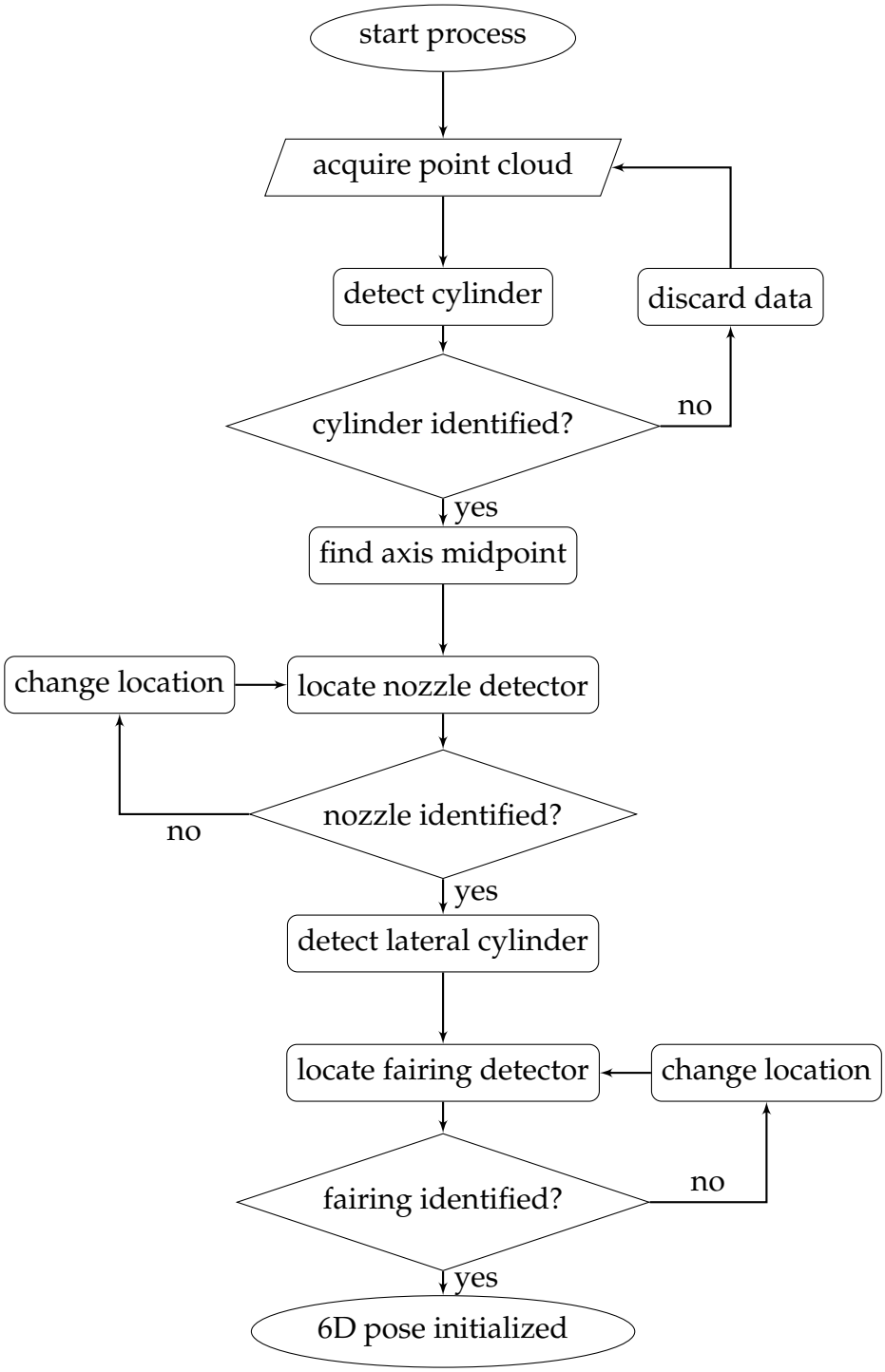


Figure 4.5. Flowchart of the initialization process using structural descriptors.

defined as:

$$\mathcal{F}_b = \mathbf{R}\mathcal{F} + \mathbf{t} \tag{4.13}$$

4.3 Tracking

Tracking is used in computer vision to follow the location of an object across multiple frames of a video stream. Under the rigid-body assumption, a subset of the point cloud –points that enter/leave the cloud, due to changes in the FOV and obstruction, are discarded– is subject to a transformation proportional to the body translation and rotation (Gómez Martínez et al., 2017).

4.3.1 Transformation Estimation

The transformation is estimated by applying the ICP algorithm (see Section 3.4) to extract the translation vector and rotation matrix that describe the inter-frame motion, finding the minimum arguments using Equation 3.23. The target cloud is the point cloud which has been initialized, using either the global descriptor matching, or the body geometry methods, observed at a time t. Consequently, the source cloud is the immediate subsequent point set at the epoch $t + \Delta t$, where Δt is the time between frames.

In order to accelerate the execution of the ICP procedure, both point clouds, target and source, are downsampled by means of voxels (see Section 2.3). The voxel filtering is also a useful method to reduce the number of points in a cloud. Here, each voxel encapsulates some points of the cloud –based on the side length of each box–, and the average point of each box will be a new member for the downsampled cloud.

After this data reduction is performed, all the elements of the cloud are then called “keypoints”, and each of them correspond to a voxel. Afterwards, a selected local descriptor (discussed in Section 3.2.1) is computed to each of the keypoints. Of course, the descriptor selection is based on the user’s requirements. After having two new sets of descriptors for each point cloud, they are compared to each other in order to find the closest match in both point sets, associating both signatures. This couple association is called as descriptor correspondence (see Figure 4.6).

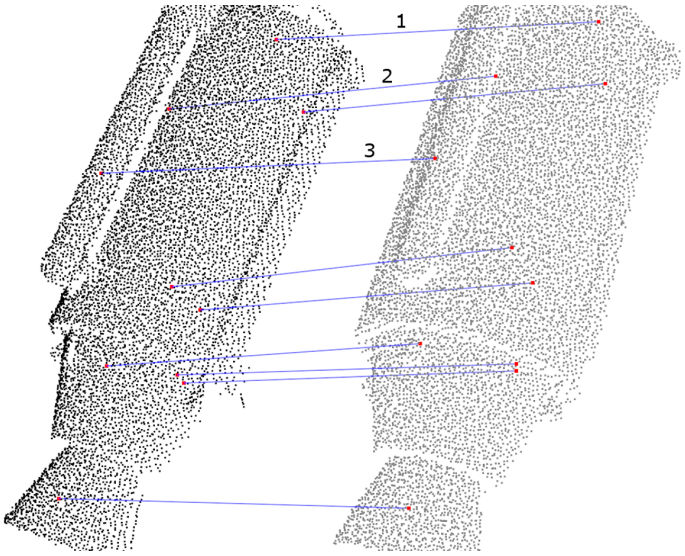


Figure 4.6. Descriptor correspondences between two consecutive views. The correspondences couples 1, 2, and 3 indicate a good feature matching. The non-labeled correspondences show wrong match due to the symmetry and the surface curvature of the object, which cause to have similar geometrical descriptors but at different location.

Those correspondences are the foundations for a first coarse registration using the SVD method (see Algorithm 3.4), but taking into account only the keypoints, whose descriptors have mutual equivalence. Hereafter, the original source cloud, i.e. that before downsampling, is then transformed using the rotation matrix and translation vector resultant from the SVD procedure, as follows:

$$\mathcal{F}_{b,int} = \mathbf{R}_{SVD}\mathcal{F}_{b,t+\Delta t} \tag{4.14}$$

being $\mathcal{F}_{int}$ the intermediate body pose after coarse alignment. The last step is performed in order to minimize the distance between point clouds. Henceforth, the ICP algorithm is applied on the complete point cloud, yielding a refined final registration of the source cloud onto the target cloud

$$\mathcal{F}_{b,t} = \mathbf{R}_{ICP}\mathcal{F}_{b,int} \tag{4.15}$$

Thus, the rocket body pose at the current view is defined as:

$$\mathcal{F}_{b,t+\Delta t} = \mathbf{R}_{SVD}^{T}\mathbf{R}_{ICP}^{T}\mathcal{F}_{b,t} \tag{4.16}$$

Finally, the translation vector $\mathbf{t}$ is obtained using Equation 3.26, being the rotation matrix $\mathbf{R}$ the product of the two matrices $\mathbf{R}_{SVD}^{T}$ and $\mathbf{R}_{ICP}^{T}$.

4.3.2 Extended Kalman Filter Design

Although the object's motion is tracked continuously by the ICP algorithm after the pose initialization, there exists an accumulation error. The latter, if not corrected, yields unacceptable biases when tracking the relative position and rotation after just a few rotations. However, the same procedure performed to initialize the relative pose of the target could be repeated after a given time to correct for the current estimation, thus eliminating, or largely mitigating, the accumulated bias.

The latter can be achieved by identifying and mapping distinctive points of the target structure, and estimating their relative position after a full rotation, if it exists. Hence, this set of distinctive points is used by a filter to compensate for estimation drifts during the pose determination of a tumbling target in similar technique to the concept of loop-closure in simultaneous localization and mapping (SLAM) techniques (Newman and Ho, 2005).

In order to determine the set of measurements and propagate the target's position and orientation, an Extended Kalman Filter (EKF) (Crassidis et al., 2007) is proposed. The EKF is a generalization of the Kalman Filter (KF) (Kalman, 1960), where the estimation for non-linear systems are then linearized both the dynamical process and the measurement updates around the average estimate.

4.3.2.1 Motion model

First of all, taking as base the target's state vector $\mathbf{x}$ (See Equation 4.1), the time-based version of the state vector can be settled as

$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{r}(t) \\ \dot{\mathbf{r}}(t) \\ q_s(t) \\ \mathbf{q}_v(t) \\ \boldsymbol{\omega}(t) \\ \mathbf{r}_f(t) \end{bmatrix} \tag{4.17}$$

where

- $\mathbf{r}$ is the target position vector
- $\dot{\mathbf{r}}$ is the target velocity vector
- $(q_s, \mathbf{q}_v)^T$ is the unit quaternion $\mathbf{q}$ (Equation 4.4) used to describe the target orientation, with $q_s^2 + \mathbf{q}_v^T \mathbf{q}_v = 1$
- $\boldsymbol{\omega}$ is the target angular velocity vector
- $\mathbf{r}_f$ is the position vector of a distinctive point on the target's surface, e.g., the fairing.

Position, translational velocity, rotation, and angular velocity are to be intended with respect to the sensor coordinates frame. In order to describe the body reference frame orientation, a rotation matrix defines the target's attitude in terms of unit quaternion as

$$\begin{aligned}\mathbf{R}(q_s, \mathbf{q}_v) &= (q_s^2 - \mathbf{q}_v^T \mathbf{q}_v)\mathbf{I}_3 + 2\mathbf{q}_v \mathbf{q}_v^T - 2q_s \boldsymbol{\Omega}_{\mathbf{q}_v} \\ &= \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 + q_0 q_3) & 2(q_1 q_3 - q_0 q_2) \\ 2(q_1 q_2 - q_0 q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 + q_0 q_1) \\ 2(q_1 q_3 + q_0 q_2) & 2(q_2 q_3 - q_0 q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}\end{aligned} \tag{4.18}$$

where

$$\boldsymbol{\Omega}_{\mathbf{q}_v} = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix} \tag{4.19}$$

On the other hand, the target's distinctive point, $\mathbf{b}_f$, is a constant vector in body coordinates, such that

$$\mathbf{r}_f(t) = \mathbf{R}(q_s(t), \mathbf{q}_v(t))\mathbf{b}_f + \mathbf{r}(t) \tag{4.20}$$

from which one obtains, by applying the identity to time instances t and $t + \Delta t$,

$$\mathbf{r}_f(t + \Delta t) = \underbrace{\mathbf{R}\left(q_s(t + \Delta t), \mathbf{q}_v(t + \Delta t)\right) \mathbf{R}^T\left(q_s(t), \mathbf{q}_v(t)\right)}_{\mathbf{R}_{\text{reg}}} \left(\mathbf{r}_f(t) - \mathbf{r}(t)\right) + \mathbf{r}(t + \Delta t) \tag{4.21}$$

Here, the rotation matrix $\mathbf{R}_{\text{reg}}$ refers to the incremental (relative) rotation estimated by the registration between two consecutive point clouds, i.e. the coarse and fine alignments extracted for the rigid transformation estimation, explained in Section 4.3.1.

Regarding to space applications, it is assumed constant linear velocity and angular velocity in the target's motion. If the chaser is not subject to internal or external forces, the relative motion between the chaser and the target yields the following state transition model between time t_k and t_{k+1}:

$$\mathbf{x}_{k+1} = \begin{bmatrix} \mathbf{r}_k + \dot{\mathbf{r}}_k(t_{k+1} - t_k) \\ \dot{\mathbf{r}}_k \\ q_{s,k+1} \\ \mathbf{q}_{v,k+1} \\ \boldsymbol{\omega}_k \\ \mathbf{R}(q_{s,k+1}, \mathbf{q}_{v,k+1})\mathbf{R}^T(q_{s,k}, \mathbf{q}_{v,k})(\mathbf{r}_{f,k} - \mathbf{r}_k) + \mathbf{r}_{k+1} \end{bmatrix} \tag{4.22}$$

where the quaternion components $q_{s,k+1}$ and $\mathbf{q}_{v,k+1}$ are obtained, for constant angular velocity $\boldsymbol{\omega}_k$ between t_k and t_{k+1}, as

$$\begin{aligned} q_{s,k+1} &= q_{s,k} \cos\left(\|\boldsymbol{\omega}_k\| \frac{(t_{k+1} - t_k)}{2}\right) + \frac{\mathbf{q}_{v,k}^T \boldsymbol{\omega}_k}{\|\boldsymbol{\omega}_k\|} \sin\left(\|\boldsymbol{\omega}_k\| \frac{(t_{k+1} - t_k)}{2}\right) \\ \mathbf{q}_{v,k+1} &= \mathbf{q}_{v,k} \cos\left(\|\boldsymbol{\omega}_k\| \frac{(t_{k+1} - t_k)}{2}\right) \\ &\quad + \frac{1}{\|\boldsymbol{\omega}_k\|} [\boldsymbol{\Omega}_{\boldsymbol{\omega}_k} \mathbf{q}_{v,k} - q_{s,k} \boldsymbol{\omega}_k] \sin\left(\|\boldsymbol{\omega}_k\| \frac{(t_{k+1} - t_k)}{2}\right) \end{aligned} \tag{4.23}$$

with skew-symmetric matrix

$$\boldsymbol{\Omega}_{\boldsymbol{\omega}_k} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{4.24}$$

Equations 4.22 and 4.23 enable one to predict the pose of the target at time t_{k+1} given an initial state at time t_k. Hence, the propagation of uncertainty stems from the linearization of the functions $\mathbf{x}_{k+1} = f_k(\mathbf{x}_k)$ as

$$\mathbf{P}_{k+1} = \mathbf{J}_{f_k} \mathbf{P}_k \mathbf{J}_{f_k}^T \tag{4.25}$$

with the Jacobian matrix $\mathbf{J}_{f_k}$ is equal to

$$\mathbf{J}_{f_k} = \begin{bmatrix} \mathbf{I}_3 & (t_{k+1}-t_k)\mathbf{I}_3 & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_3 & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & \frac{\partial q_{s,k+1}}{\partial q_{s,k}} & \frac{\partial q_{s,k+1}}{\partial q_{v,k}} & \frac{\partial q_{s,k+1}}{\partial \omega_k} & \mathbf{0}_{1\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \frac{\partial q_{v,k+1}}{\partial q_{s,k}} & \frac{\partial q_{v,k+1}}{\partial q_{v,k}} & \frac{\partial q_{v,k+1}}{\partial \omega_k} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times3} & \mathbf{I}_3 & \mathbf{0}_{3\times3} \\ \mathbf{I}_3-\mathbf{R}(q_{s,k+1},q_{v,k+1})\mathbf{R}^T(q_{s,k},q_{v,k}) & (t_{k+1}-t_k)\mathbf{I}_3 & \frac{\partial r_{f,k+1}}{\partial q_{s,k}} & \frac{\partial r_{f,k+1}}{\partial q_{v,k}} & \frac{\partial r_{f,k+1}}{\partial \omega_k} & \mathbf{R}(q_{s,k+1},q_{v,k+1})\mathbf{R}^T(q_{s,k},q_{v,k}) \end{bmatrix} \tag{4.26}$$

having the correspondent partial derivatives

$$\begin{aligned} \frac{\partial q_{s,k+1}}{\partial q_{s,k}} &= \cos\left(\|\omega_k\|\frac{(t_{k+1}-t_k)}{2}\right) \\ \frac{\partial q_{s,k+1}}{\partial q_{v,k}} &= \frac{\omega_k^T}{\|\omega_k\|}\sin\left(\|\omega_k\|\frac{(t_{k+1}-t_k)}{2}\right) \\ \frac{\partial q_{s,k+1}}{\partial \omega_k} &= \frac{-1}{\|\omega_k\|}\left[\frac{q_{v,k}^T\Omega_{\omega_k}^2}{\|\omega_k\|^2}+\frac{(t_{k+1}-t_k)}{2}q_{s,k}\omega_k^T\right]\sin\left(\|\omega_k\|\frac{(t_{k+1}-t_k)}{2}\right) \\ &\quad + q_{v,k}^T\frac{\omega_k\omega_k^T}{\|\omega_k\|^2}\frac{(t_{k+1}-t_k)}{2}\cos\left(\|\omega_k\|\frac{(t_{k+1}-t_k)}{2}\right) \end{aligned} \tag{4.27}$$

$$
\begin{aligned}
\frac{\partial \mathbf{q}_{v,k+1}}{\partial q_{s,k}} &= -\frac{\boldsymbol{\omega}_k}{\|\boldsymbol{\omega}_k\|} \sin\left(\|\boldsymbol{\omega}_k\| \frac{(t_{k+1}-t_k)}{2}\right) \\
\frac{\partial \mathbf{q}_{v,k+1}}{\partial \mathbf{q}_{v,k}} &= \begin{bmatrix} q_{1,k} & 0 & 0 \\ 0 & q_{2,k} & 0 \\ 0 & 0 & q_{3,k} \end{bmatrix} \cos\left(\|\boldsymbol{\omega}_k\| \frac{(t_{k+1}-t_k)}{2}\right) \\
&\quad + \frac{1}{\|\boldsymbol{\omega}_k\|} \boldsymbol{\Omega}_{\omega_k} \sin\left(\|\boldsymbol{\omega}_k\| \frac{(t_{k+1}-t_k)}{2}\right) \\
\frac{\partial \mathbf{q}_{v,k+1}}{\partial \boldsymbol{\omega}_k} &= \frac{(t_{k+1}-t_k)}{2\|\boldsymbol{\omega}_k\|^2} \left[\boldsymbol{\Omega}_{\omega_k} \mathbf{q}_{v,k} - q_{s,k} \boldsymbol{\omega}_k\right] \boldsymbol{\omega}_k^T \cos\left(\|\boldsymbol{\omega}_k\| \frac{(t_{k+1}-t_k)}{2}\right) \\
&\quad + \frac{1}{\|\boldsymbol{\omega}_k\|} \left[-\frac{(t_{k+1}-t_k)}{2} \mathbf{q}_{v,k} \boldsymbol{\omega}_k^T - \boldsymbol{\Omega}_{\mathbf{q}_{v,k}} - q_{s,k} \begin{bmatrix} \omega_{1,k} & 0 & 0 \\ 0 & \omega_{2,k} & 0 \\ 0 & 0 & \omega_{3,k} \end{bmatrix}\right] \\
&\quad \cdot \sin\left(\|\boldsymbol{\omega}_k\| \frac{(t_{k+1}-t_k)}{2}\right) \\
&\quad - \left[\boldsymbol{\Omega}_{\omega_k} \mathbf{q}_{v,k} - q_{s,k} \boldsymbol{\omega}_k\right] \frac{\boldsymbol{\omega}_k^T}{\|\boldsymbol{\omega}_k\|^3} \sin\left(\|\boldsymbol{\omega}_k\| \frac{(t_{k+1}-t_k)}{2}\right)
\end{aligned}
\tag{4.28}
$$

$$
\begin{aligned}
\frac{\partial \mathbf{r}_{f,k+1}}{\partial q_{s,k}} &= 2(q_{s,k+1}\mathbf{I}_3 - \mathbf{\Omega}_{\mathbf{q}_{v,k+1}}) \begin{pmatrix} a_k \\ b_k \\ c_k \end{pmatrix} \frac{\partial q_{s,k+1}}{\partial q_{s,k}} + \mathbf{R}(q_{s,k+1}, \mathbf{q}_{v,k+1}) \begin{pmatrix} \frac{\partial a_k}{\partial q_{s,k}} \\ \frac{\partial b_k}{\partial q_{s,k}} \\ \frac{\partial c_k}{\partial q_{s,k}} \end{pmatrix} \\
\frac{\partial \mathbf{r}_{f,1,k+1}}{\partial \mathbf{q}_{v,k}} &= 2(q_{s,k+1}, \mathbf{q}_{v,k+1}^T) \begin{bmatrix} a_k & 0 & -c_k & b_k \\ 0 & a_k & b_k & c_k \\ -c_k & b_k & -a_k & 0 \\ b_k & c_k & 0 & -a_k \end{bmatrix} \begin{pmatrix} \frac{\partial q_{s,k+1}}{\partial \mathbf{q}_{v,k}} \\ \frac{\partial \mathbf{q}_{v,k+1}}{\partial \mathbf{q}_{v,k}} \end{pmatrix} + \mathbf{r}_{1,k+1}^T \begin{pmatrix} \frac{\partial a_k}{\partial \mathbf{q}_{v,k}} \\ \frac{\partial b_k}{\partial \mathbf{q}_{v,k}} \\ \frac{\partial c_k}{\partial \mathbf{q}_{v,k}} \end{pmatrix} \\
\frac{\partial \mathbf{r}_{f,2,k+1}}{\partial \mathbf{q}_{v,k}} &= 2(q_{s,k+1}, \mathbf{q}_{v,k+1}^T) \begin{bmatrix} b_k & c_k & 0 & -a_k \\ c_k & -b_k & a_k & 0 \\ 0 & a_k & b_k & c_k \\ -a_k & 0 & c_k & -b_k \end{bmatrix} \begin{pmatrix} \frac{\partial q_{s,k+1}}{\partial \mathbf{q}_{v,k}} \\ \frac{\partial \mathbf{q}_{v,k+1}}{\partial \mathbf{q}_{v,k}} \end{pmatrix} + \mathbf{r}_{2,k+1}^T \begin{pmatrix} \frac{\partial a_k}{\partial \mathbf{q}_{v,k}} \\ \frac{\partial b_k}{\partial \mathbf{q}_{v,k}} \\ \frac{\partial c_k}{\partial \mathbf{q}_{v,k}} \end{pmatrix} \\
\frac{\partial \mathbf{r}_{f,3,k+1}}{\partial \mathbf{q}_{v,k}} &= 2(q_{s,k+1}, \mathbf{q}_{v,k+1}^T) \begin{bmatrix} c_k & -b_k & a_k & 0 \\ -b_k & -c_k & 0 & a_k \\ a_k & 0 & -c_k & b_k \\ 0 & a_k & b_k & c_k \end{bmatrix} \begin{pmatrix} \frac{\partial q_{s,k+1}}{\partial \mathbf{q}_{v,k}} \\ \frac{\partial \mathbf{q}_{v,k+1}}{\partial \mathbf{q}_{v,k}} \end{pmatrix} + \mathbf{r}_{3,k+1}^T \begin{pmatrix} \frac{\partial a_k}{\partial \mathbf{q}_{v,k}} \\ \frac{\partial b_k}{\partial \mathbf{q}_{v,k}} \\ \frac{\partial c_k}{\partial \mathbf{q}_{v,k}} \end{pmatrix} \\
\frac{\partial \mathbf{r}_{f,1,k+1}}{\partial \boldsymbol{\omega}_k} &= 2(q_{s,k+1}, \mathbf{q}_{v,k+1}^T) \begin{bmatrix} a_k & 0 & -c_k & b_k \\ 0 & a_k & b_k & c_k \\ -c_k & b_k & -a_k & 0 \\ b_k & c_k & 0 & -a_k \end{bmatrix} \begin{pmatrix} \frac{\partial q_{s,k+1}}{\partial \boldsymbol{\omega}_k} \\ \frac{\partial \mathbf{q}_{v,k+1}}{\partial \boldsymbol{\omega}_k} \end{pmatrix} \\
\frac{\partial \mathbf{r}_{f,2,k+1}}{\partial \boldsymbol{\omega}_k} &= 2(q_{s,k+1}, \mathbf{q}_{v,k+1}^T) \begin{bmatrix} b_k & c_k & 0 & -a_k \\ c_k & -b_k & a_k & 0 \\ 0 & a_k & b_k & c_k \\ -a_k & 0 & c_k & -b_k \end{bmatrix} \begin{pmatrix} \frac{\partial q_{s,k+1}}{\partial \boldsymbol{\omega}_k} \\ \frac{\partial \mathbf{q}_{v,k+1}}{\partial \boldsymbol{\omega}_k} \end{pmatrix} \\
\frac{\partial \mathbf{r}_{f,3,k+1}}{\partial \boldsymbol{\omega}_k} &= 2(q_{s,k+1}, \mathbf{q}_{v,k+1}^T) \begin{bmatrix} c_k & -b_k & a_k & 0 \\ -b_k & -c_k & 0 & a_k \\ a_k & 0 & -c_k & b_k \\ 0 & a_k & b_k & c_k \end{bmatrix} \begin{pmatrix} \frac{\partial q_{s,k+1}}{\partial \boldsymbol{\omega}_k} \\ \frac{\partial \mathbf{q}_{v,k+1}}{\partial \boldsymbol{\omega}_k} \end{pmatrix}
\end{aligned}
\tag{4.29}
$$

where $\mathbf{R}(q_{s,k+1}, \mathbf{q}_{v,k+1}) = \begin{bmatrix} \mathbf{r}_{1,k+1}^T \\ \mathbf{r}_{2,k+1}^T \\ \mathbf{r}_{3,k+1}^T \end{bmatrix}$ and a_k, b_k, c_k (and corresponding derivatives)

$$
\begin{aligned}
(a_k, b_k, c_k)^T &= \mathbf{R}^T(q_{s,k}, \mathbf{q}_{v,k})(\mathbf{r}_{f,k} - \mathbf{r}_k) \\
\left(\frac{\partial a_k}{\partial q_{s,k}}, \frac{\partial b_k}{\partial q_{s,k}}, \frac{\partial c_k}{\partial q_{s,k}}\right)^T &= 2\left[q_{s,k}\mathbf{I}_3 + \mathbf{\Omega}_{\mathbf{q}_{v,k}}\right](\mathbf{r}_{f,k} - \mathbf{r}_k) \\
\frac{\partial a_k}{\partial \mathbf{q}_{v,k}} &= 2(\mathbf{r}_{f,k} - \mathbf{r}_k)^T \begin{bmatrix} q_{1,k} & -q_{2,k} & -q_{3,k} \\ q_{2,k} & q_{1,k} & -q_{0,k} \\ q_{3,k} & q_{0,k} & q_{1,k} \end{bmatrix} \\
\frac{\partial b_k}{\partial \mathbf{q}_{v,k}} &= 2(\mathbf{r}_{f,k} - \mathbf{r}_k)^T \begin{bmatrix} q_{2,k} & q_{1,k} & q_{0,k} \\ -q_{1,k} & q_{2,k} & -q_{3,k} \\ -q_{0,k} & q_{3,k} & q_{2,k} \end{bmatrix} \\
\frac{\partial c_k}{\partial \mathbf{q}_{v,k}} &= 2(\mathbf{r}_{f,k} - \mathbf{r}_k)^T \begin{bmatrix} q_{3,k} & -q_{0,k} & q_{1,k} \\ q_{0,k} & q_{3,k} & q_{2,k} \\ -q_{1,k} & -q_{2,k} & q_{3,k} \end{bmatrix}
\end{aligned}
\tag{4.30}
$$

Although the complexity of the Jacobian is noticeable, it allows to propagate the attitude of the target in time without taking into account the inertia properties of the object. This is crucial, because bad assumptions about the inertia tensor values can produce unexpected results in the motion estimation process.

4.3.2.2 State measurement

At each consecutive frame, an observation of the position and orientation of the target is provided via the transformation estimation (see Section 4.3.1),

$$
\mathcal{E}(\mathbf{z}_{k+1}) = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times4} & \mathbf{0}_{3\times6} \\ \mathbf{0}_{4\times3} & \mathbf{0}_{4\times3} & \mathbf{I}_4 & \mathbf{0}_{4\times6} \end{bmatrix} \mathbf{x}_{k+1} = \mathbf{H}_{k+1}\mathbf{x}_{k+1} \tag{4.31a}
$$

$$
\mathcal{D}(\mathbf{z}_{k+1}) = \mathbf{R}_{z,k+1} \tag{4.31b}
$$

where $\mathcal{E}(\cdot)$ and $\mathcal{D}(\cdot)$ denote the expectation (mean state value) and dispersion (covariance) operator, respectively. When the distinctive point on the target $\mathbf{r}_f$ is re-observed, the observation model (4.31a) modifies as

$$
\mathcal{E}(\mathbf{z}_{k+1}) = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times4} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{4\times3} & \mathbf{0}_{4\times3} & \mathbf{I}_4 & \mathbf{0}_{4\times3} & \mathbf{0}_{4\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times4} & \mathbf{0}_{3\times3} & \mathbf{I}_3 \end{bmatrix} \mathbf{x}_{k+1} = \mathbf{H}'_{k+1}\mathbf{x}_{k+1} \tag{4.32}
$$

Thereby, it is possible to execute the prediction and update steps of the EKF can then be formulated as

$$\begin{aligned}
&\text{Prediction:}\\
&\quad \bar{\mathbf{x}}_{k+1} = f_k(\mathbf{x}_k)\\
&\quad \bar{\mathbf{P}}_{k+1} = \mathbf{J}_{f_k}\mathbf{P}_k\mathbf{J}_{f_k}^T + \mathbf{Q}_k\\
&\text{Update:}\\
&\quad \mathbf{K}_{k+1} = \bar{\mathbf{P}}_{k+1}\mathbf{H}_{k+1}^T\left(\mathbf{H}_{k+1}\bar{\mathbf{P}}_{k+1}\mathbf{H}_{k+1}^T + \mathbf{R}_{z,k+1}\right)^{-1}\\
&\quad \mathbf{x}_{k+1} = \bar{\mathbf{x}}_{k+1} + \mathbf{K}_{k+1}\left(\mathbf{z}_{k+1} - \mathbf{H}_{k+1}\bar{\mathbf{x}}_{k+1}\right)\\
&\quad \mathbf{P}_{k+1} = \left(\mathbf{I}_3 - \mathbf{K}_{k+1}\mathbf{H}_{k+1}\right)\bar{\mathbf{P}}_{k+1}
\end{aligned} \tag{4.33}$$

where $\mathbf{Q}_k$ denotes the process noise added at each step to model the uncertainty about the motion model employed. The process noise added is

$$\mathbf{Q}_k = \mathbf{J}_{f_k}\begin{bmatrix}
\mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times4} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3}\\
\mathbf{0}_{3\times3} & \sigma_v^2\mathbf{I}_3 & \mathbf{0}_{3\times4} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3}\\
\mathbf{0}_{4\times3} & \mathbf{0}_{4\times3} & \mathbf{0}_{4\times4} & \mathbf{0}_{4\times3} & \mathbf{0}_{4\times3}\\
\mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times4} & \sigma_\omega^2\mathbf{I}_3 & \mathbf{0}_{3\times3}\\
\mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times4} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3}
\end{bmatrix}\mathbf{J}_{f_k}^T \tag{4.34}$$

being σ_v and σ_ω the linear velocity and angular velocity standard deviations, respectively.

Figure 4.7 illustrates the EKF flowchart: after initialization, the state vector at the next frame is predicted and updated via the assumed motion model and the rigid transformation estimation given in Section 4.3.1. When the chaser senses one full rotation of the target, or after a given fixed elapsed time, a re-initialization to detect one distinctive point on the target surface (given in Section 4.2) is triggered. The latter, as can be remembered, enables a correction for the drift in the estimated target pose.

4.4 Summary

The current chapter introduced the paradigm of the pose estimation for a target tumbling freely in orbit. After the establishment of certain orbital conditions, the

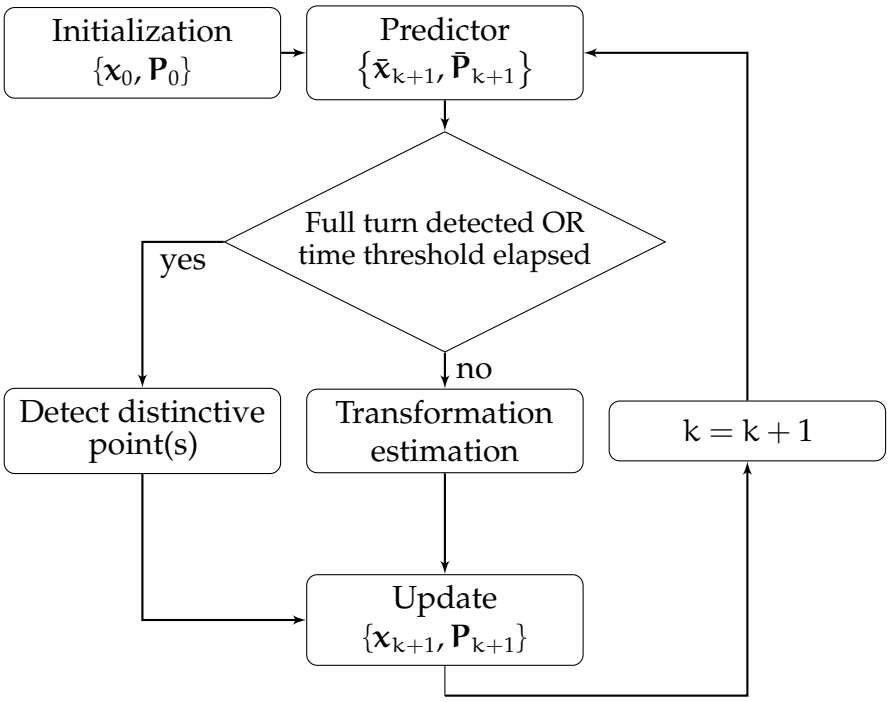


Figure 4.7. EKF process flowchart.

presentation of the target's state vector was given. The pose estimation problem is composed by two phases, which are the pose initialization and the target's tracking. The former indicated the procedures to obtain the relative position and orientation of an object having the sensor coordinates frame as the reference one. Two different approaches for initialization procedure were exposed, using geometrical descriptors with offline training, or using geometric structural information. On the other hand, the tracking phase was explained, whose main objective is to estimate the motion of the target using consecutive point clouds. This procedure included how to find the rigid transformation between clouds, and to use the EKF in order to have a better estimation of the target's state parameters. The complete definition of the filter was indicated.

5 | Simulations and Results

Two simulations were performed to assess the pose estimation process exposed in Chapter 4. Firstly, the synthetic simulation of the target and the point cloud data, and secondly, the use of a scaled model with a real ToF camera, previously described in Section 2.1.2.

The synthetic simulations were performed using a CAD model of the Cosmos-3M upper stage (see Figure 1.4), whose dimensions were based on public available data, and assumed as true. The purpose of using the synthetic data was to validate the performance and outcome of the algorithms during the development phase of the project. In principle, the point clouds did not include any level of noise. In a later phase, Gaussian noise was added in order to have a better representation of possible real data.

On the other hand, the simulations using the real sensor were made in order to evaluate the robustness of the algorithms in real world, because the noise in simulated data differs from the real noised data. Here, the model used was built using a 3D printer, and in a scale of 1:10. The latter forced to modify the initial algorithm, scaling the internal parameters in the same proportion, as well as the distance between the sensor and the observed object. However, the illumination conditions that a target may have in orbit were not taken into account for the algorithm validation.

5.1 Synthetic Data

For algorithm prototyping and validation of the first outcomes, the use of synthetic data was crucial for the initial development phase of the project. First of all, the construction of the CAD rocket body based on public data was assumed to be

true. Furthermore, the production of synthetic point clouds was possible thanks to BlenSor, as mentioned in Section 4.2.1.1.

5.1.1 Pose Initialization using Global Descriptors

First of all, the generation of a training database was necessary in order to have a repository of descriptors, when a signature match is executed in order to find the nearest neighbor. Studies have shown that rocket bodies in LEO may tumble with an angular velocity up to $10°/s$ (Ojakangas and Cowardin, 2012; Yanagisawa and Kurosaki, 2012). Hence, a stack of point clouds were generated, where the rocket body was rotating at the same angular velocity described previously, i.e. $10°/s$, in order to have an extreme rotational state.

However, the rotation of the target may provoke the orientation of the latter to be in any direction. Thus, the point clouds created were covered for the whole sphere, with an angle step between positions of $10°$ with respect to the x- and y-axis of the model reference frame. Then, a rotation through the z-axis was carried until the target returned to the initial position, but without recording the last orientation again, to avoid duplicates. At the end, the total number of point clouds generated using this procedure was 22104, i.e. the number of descriptors stored in the training database. At this point, there was no noise added to the point clouds.

Before generating the global descriptors, a robustness analysis was performed in order to see which type of signature could offer more information in each histogram, avoiding the possible false positives due to the symmetry of the target. Figures 3.9, 3.11, and 3.12 shows an example of the three global descriptors evaluated –VFH, CVFH, and OUR-CVFH– for the same rocket body's orientation. At the end, the use of OUR-CVFH was chosen due to the additional information that the descriptor has due to the local reference frames created after the point cloud clustering (see Section 3.2.2.3). Figure 5.1 shows an arbitrary point cloud view of the rocket body, meanwhile Figure 5.2 presents the two closest matches of the test view on the training list, using the adopted global descriptor.

Although the possible candidates for matching the pose of the test point cloud is feasible, the matching process sometimes presented many false positives, i.e. that the closest neighbors found in the histogram had different orientation to the outcome expected (see right part of Figure 5.2). The main reason behind this problem is the

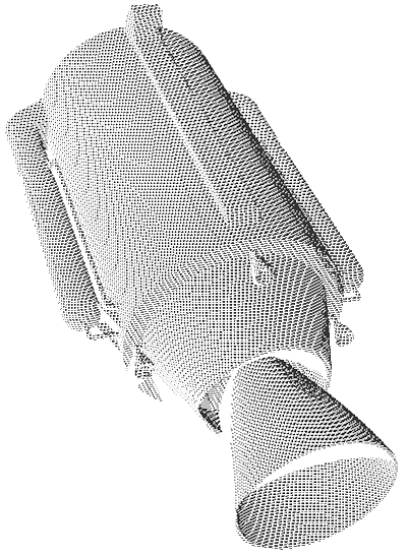


Figure 5.1. Test point cloud for evaluation of pose initialization using a global descriptor.

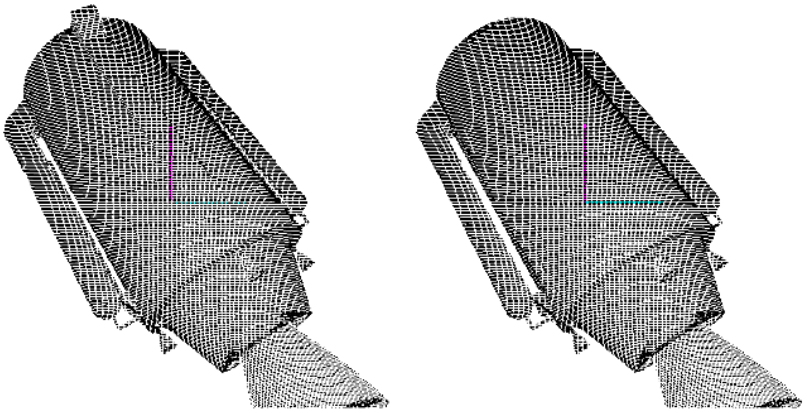
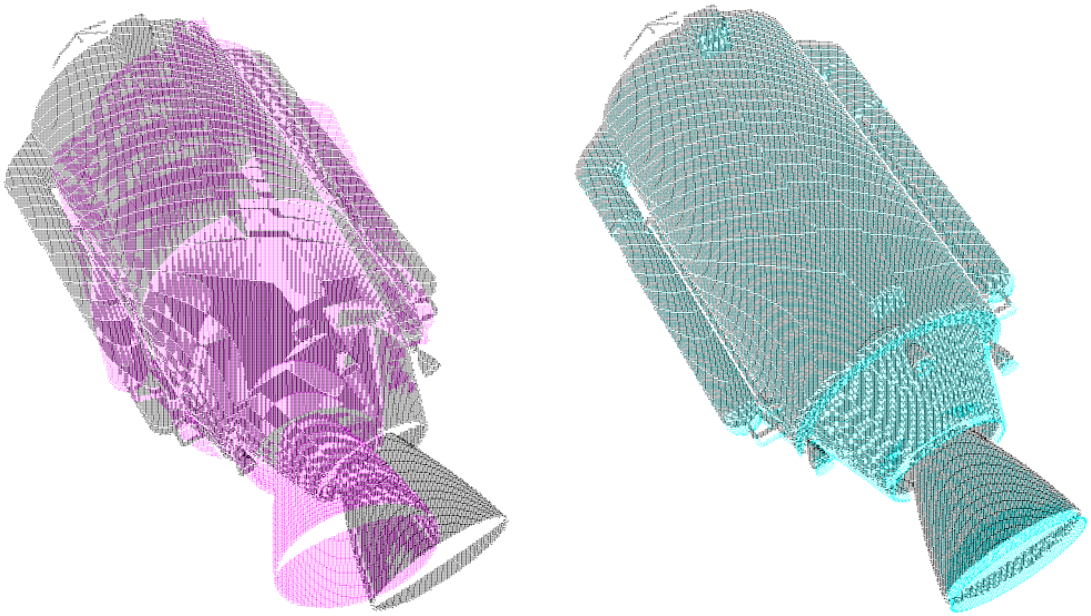


Figure 5.2. Closest candidates to match the test view. The comparison of the test view global descriptor and the trained descriptors is performed by finding the minimum χ^2 distance between them, using brute force, i.e. per element in the training list.



(a) Test view (source, magenta) and best candidate (target, black) point clouds before ICP.

(b) Clouds alignment after ICP. Transformed cloud (source, cyan) and best candidate (target, black).

Figure 5.3. Pose initialization using ICP algorithm after finding best matching candidate. The registration is not completely accurate: a longitudinal mismatch is noticeable at the nozzle's end (5.3b).

symmetry of the rocket body, despite of having external appendages that may help to distinguish between the aft and the front of the rocket body.

When the certainty about the closest match for the current body pose is assured, then it is possible to apply the ICP algorithm to register the point clouds and find the transformation matrix, i.e. rotation and translation, between the current view and the database candidate. As a result, the test view pose is the product of the match pose and the rotation matrix from the ICP. At the time of this analysis, no improvement was made to the registration process, in comparison to that explained in Section 4.3.1. Outcome of the ICP procedure is shown in Figure 5.3.

As a final remark, it is important to take into account the condition of the target's surface for real operation: if the object has a variation in its structure, there would not be any possibility for a descriptor matching. Unfortunately, the training database assumed the complete integrity of the rocket body. If this completeness cannot be assured, the procedure would not work as expected.

5.1.2 Pose Initialization based on Body Geometry

For this assessment, there was not required a full stack of different point clouds, but only one. A initial pose was given for debugging the algorithm and evaluating its effectiveness. Table 5.1 reports the conditions of a stable rotation of the rocket body, where the latter tumbles about an axis parallel to the sensor's y-axis. Using 37 point clouds, i.e. one full rotation, the initialization process was performed based on the schemes described in Section 4.2.2. Figure 5.4 is a feasible result of the process of initialization.

Table 5.1. Synthetic Data Simulation - Body Initial Parameters

Parameter	Value
Roll angle (about sensor's x-axis)	$-90°$
Pitch angle (about sensor's y-axis)	$-70°$
Yaw angle (about sensor's z-axis)	$-20°$
Rotation axis in sensor reference frame	$[0, 1, 0]$
Angular velocity	$10°/s$
Sensor acquisition frequency	1Hz
Number of simulated frames	100

Each point cloud was processed independently to obtain the target's pose. Figure 5.5 shows the difference between the nominal and the estimated attitude of the rocket body, given in Euler angles. The attitude was correctly retrieved only when the fairing was in the FOV of the sensor, i.e. 31 out of 37 point clouds. Each Euler angle –roll, pitch and yaw– could be initialized with an error not grater than $2°$, always when the fairing was visible (see Figure 5.6). Hence, the routine is not capable of estimate the 6DOF pose in the absence of this structural appendage. This flaw can be solved using analysis of 2D image features in addition to the 3D data, helping to identify another visual clues on the rocket's surface. On the other hand, if there are no such keypoints or the target's surface is too much degraded due to the outer space environment, the probabilities of enhancing this method are considerably reduced.

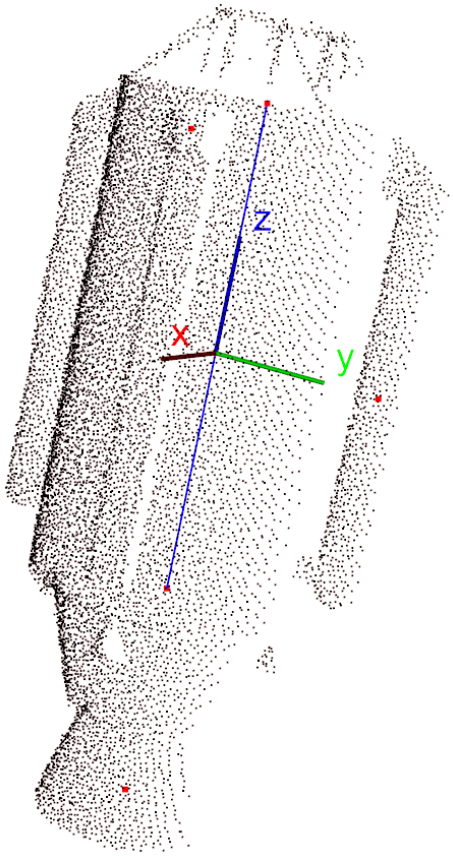


Figure 5.4. Outcome of the initialization process based on body geometry. The body reference frame is located at the centroid of the cylinder. The cylinder's axis of symmetry is shown in blue, meanwhile the location for the axis' endpoints, as well as the location for nozzle and fairing detectors are shown in red.

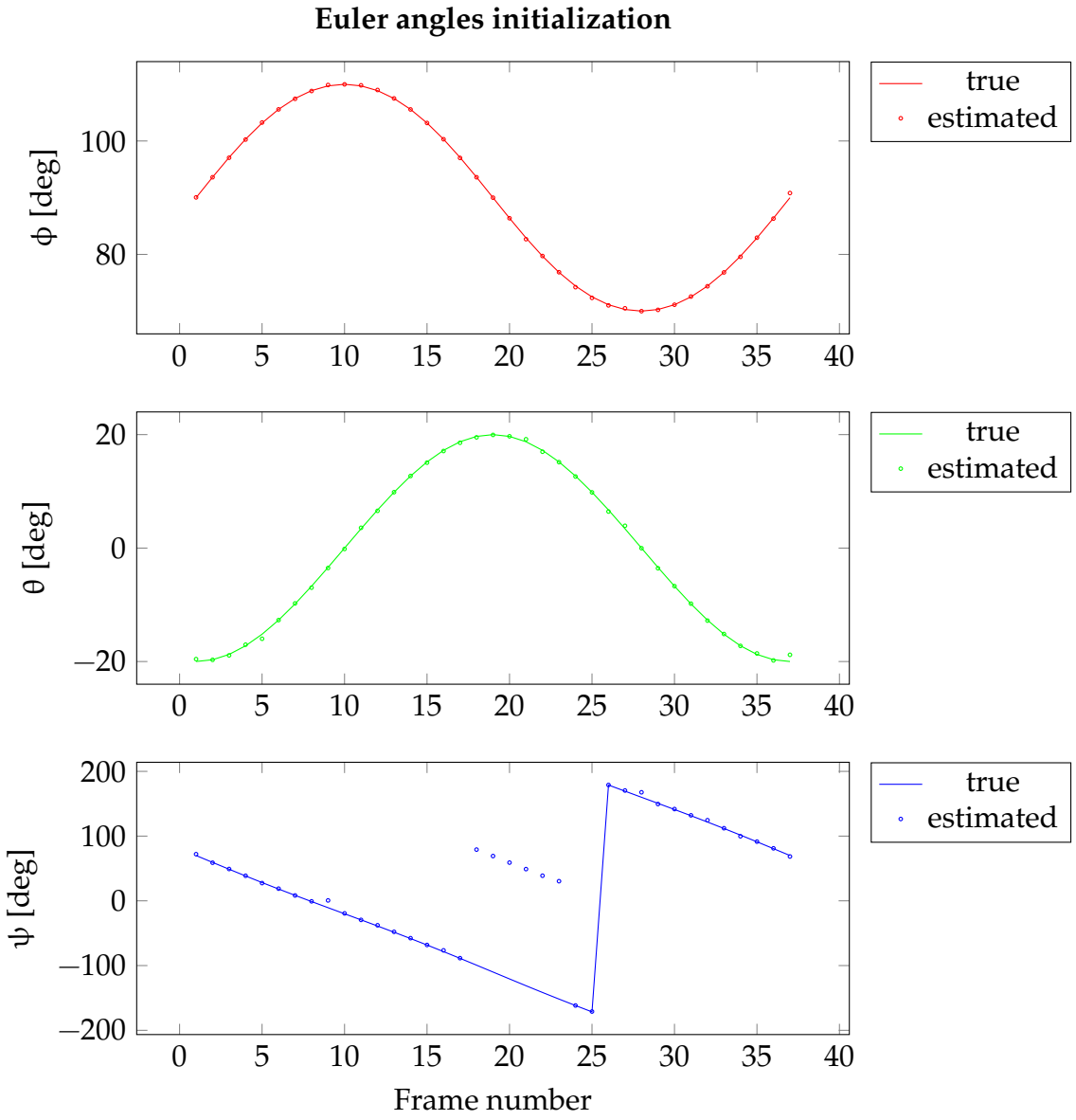


Figure 5.5. Output of the initialization routine for estimating the rocket orientation. Both estimated and nominal (true) Euler angles are given. The six incorrect initializations of the rocket yaw angle are due to the lack of view of the fairing from the sensor, which could not resolve rotations about the rocket's longitudinal axis.

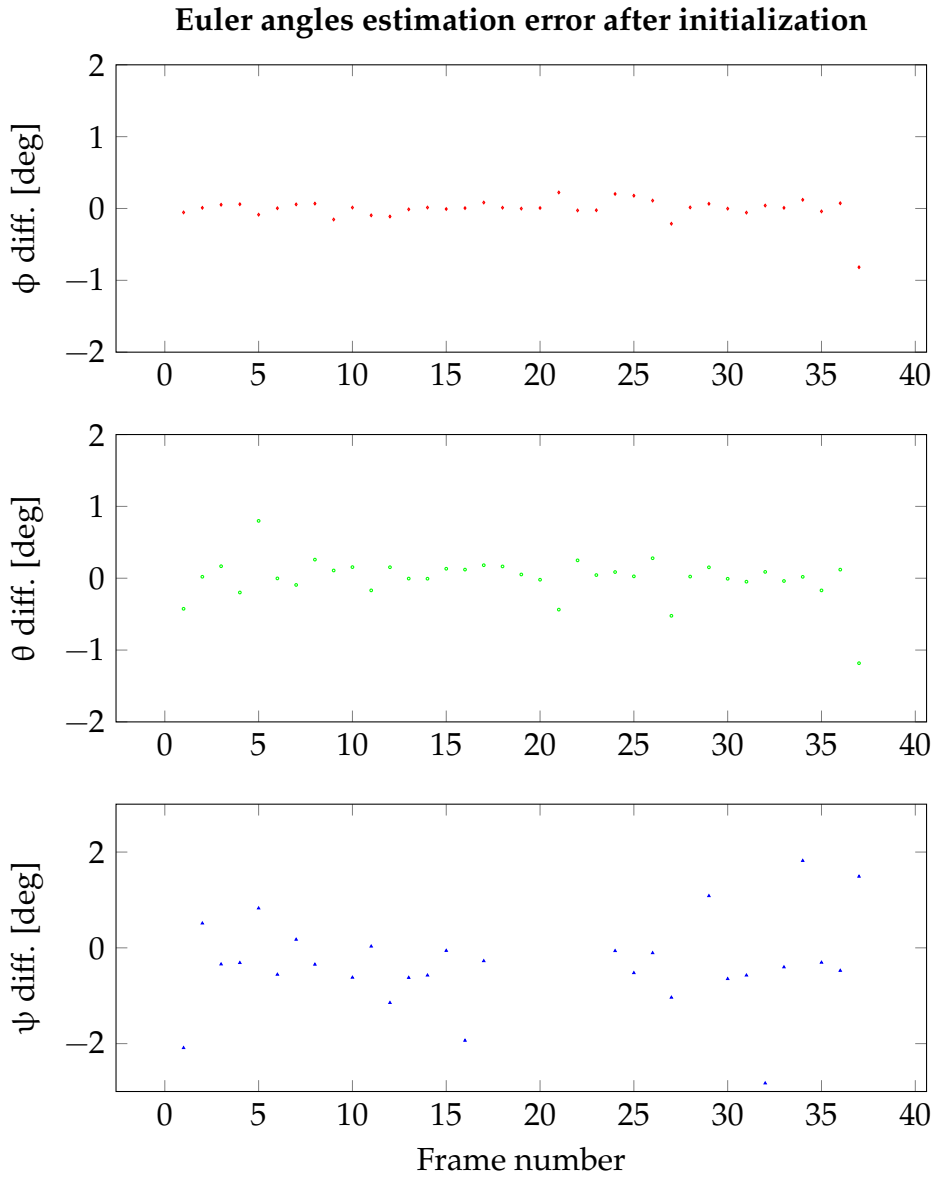


Figure 5.6. Difference between the estimated and true relative rocket orientations, given in Euler angles, following the initialization procedure.

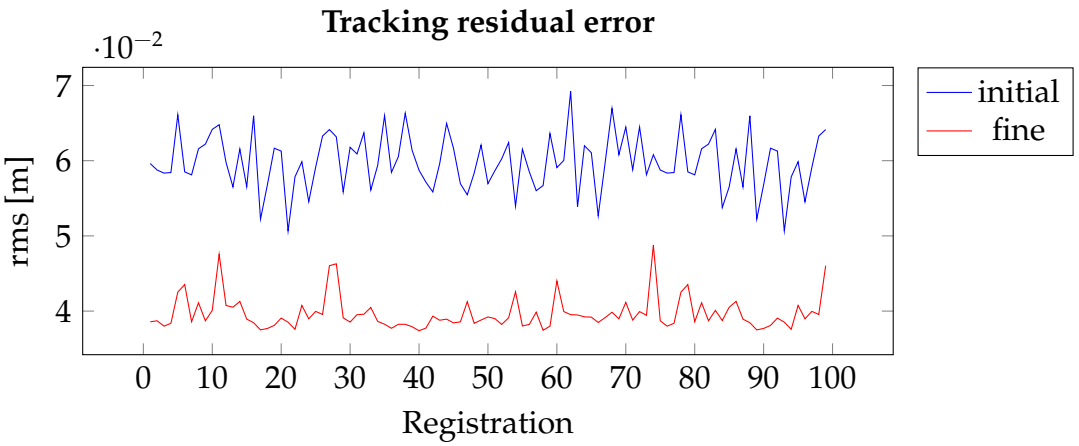


Figure 5.7. Residual error for the alignment of each pair of consecutive frames. The blue curve shows the coarse alignment using SVD. The red curve shows the fine alignment using ICP, after the coarse one.

5.1.3 Pose Tracking

After the pose was initialized, the motion tracking of the target can start. First, the registration process of two consecutive point clouds was necessary to obtain the rigid transformation between them, using the procedure explained in Section 4.3.1. From the initial conditions exposed in Table 5.1, 100 point clouds were generated, representing 2.78 full 360°-rotations. Gaussian white noise was added to the simulated point cloud: each point was given an uncertainty of 1cm (standard deviation).

Figure 5.7 shows the residual errors related to the alignment of two consecutive point clouds, where the coarse and fine alignment had been involved. The enhancement obtained when passing from the coarse alignment (where the point cloud was downsampled to keypoints) to the refined estimation (when the linear and angular distance were reduced thanks to the initial alignment, but using the whole number of points in the cloud) is noteworthy.

The residual rms error is not exceeding 7cm, provoked by the non-perfect association of the keypoints correspondences, additional to the non-equality of the point data sets. If some visual markers could be added to the initial coarse alignment and their correspondent feature descriptors, the accuracy of the initial registration would improve considerably. When the fine alignment took part by using the ICP, the error is reduced, not being greater than 5cm. However, there were some local drawbacks, when the two point clouds possessed a certain degree of symmetry.

Figure 5.8. Larger registration errors due to an erroneous rotation alignment about the rocket longitudinal axis, noticeable by the displacement of the lateral tank. The point clouds are seen from the nozzle viewpoint, with the current point cloud in gray and the former point cloud in black.

The latter might produce a larger alignment error on a target's rotation close to the rocket's longitudinal axis. Figure 5.8 shows an example of this situation, where the offset in the transverse section of the body becomes noticeable.

When the rigid transformation was obtained, the estimation of the target's state vector was the following step in order to predict and update the rocket body's pose via the EKF, exposed in Section 4.3.2. The EKF required as initial values the position, linear velocity, orientation and angular velocity in the sensor frame. All of them were obtained from the pose initialization as well as the rigid transformation estimation (initial step of the tracking phase). The EKF converged rapidly towards the correct pose when the correct initialization was introduced and the registration did not fail. Figure 5.9 shows the true and estimated values for the rocket body's state vector, meanwhile Figure 5.10 shows the error between the estimated values by the EKF and the ground truth values for the state vector.

On the other hand, the valid estimation of the relative orientation of the rocket body is crucial for the capture phase of the autonomous rendezvous. Hence, Figure 5.11 visualizes the true and estimated values of the orientation's Euler angles. Additionally, the accuracy in the tracking of the rocket orientation is shown in Fi-

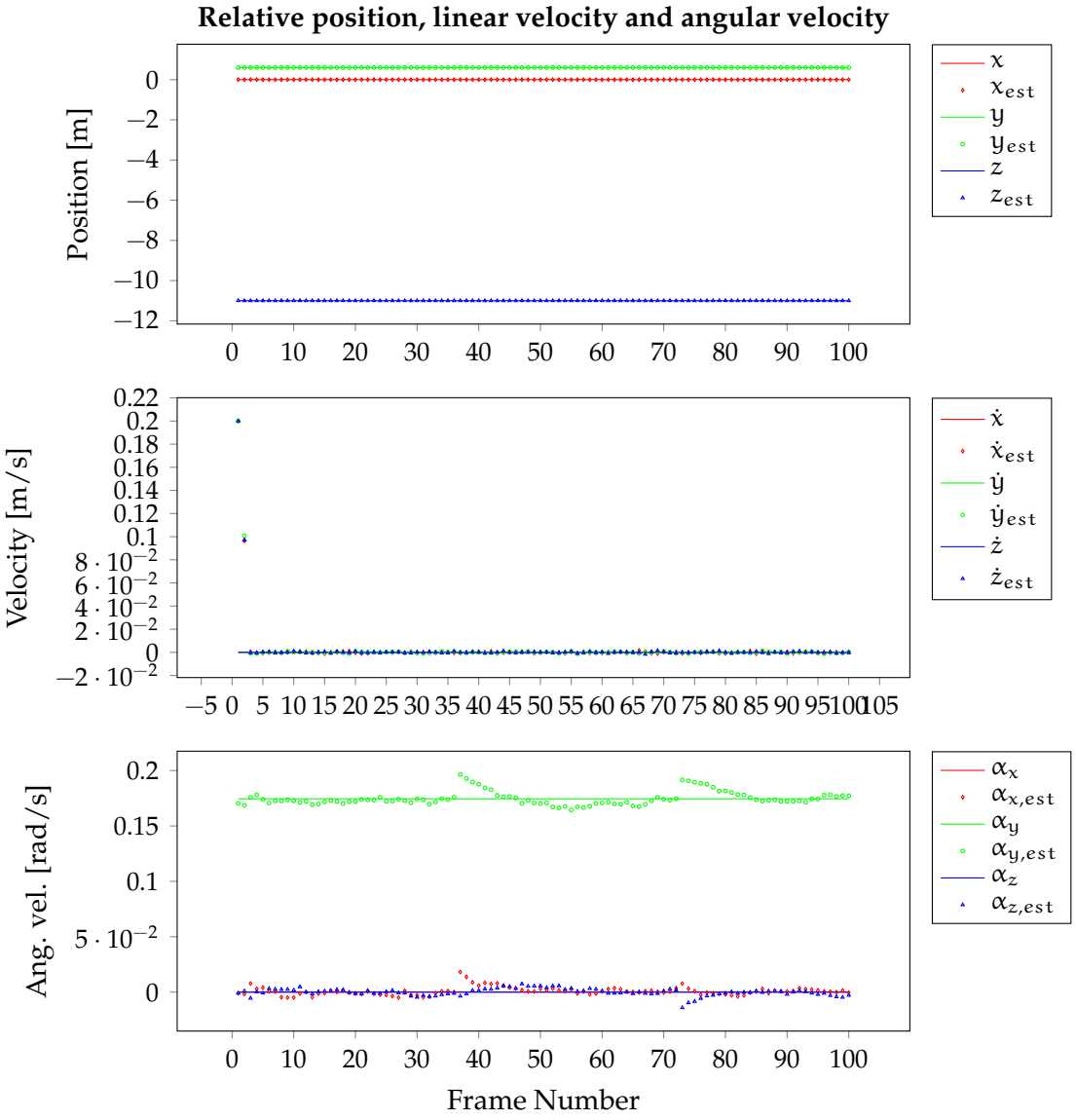


Figure 5.9. Comparison of the true and estimated relative position, linear velocity, and angular velocity output by the EKF.

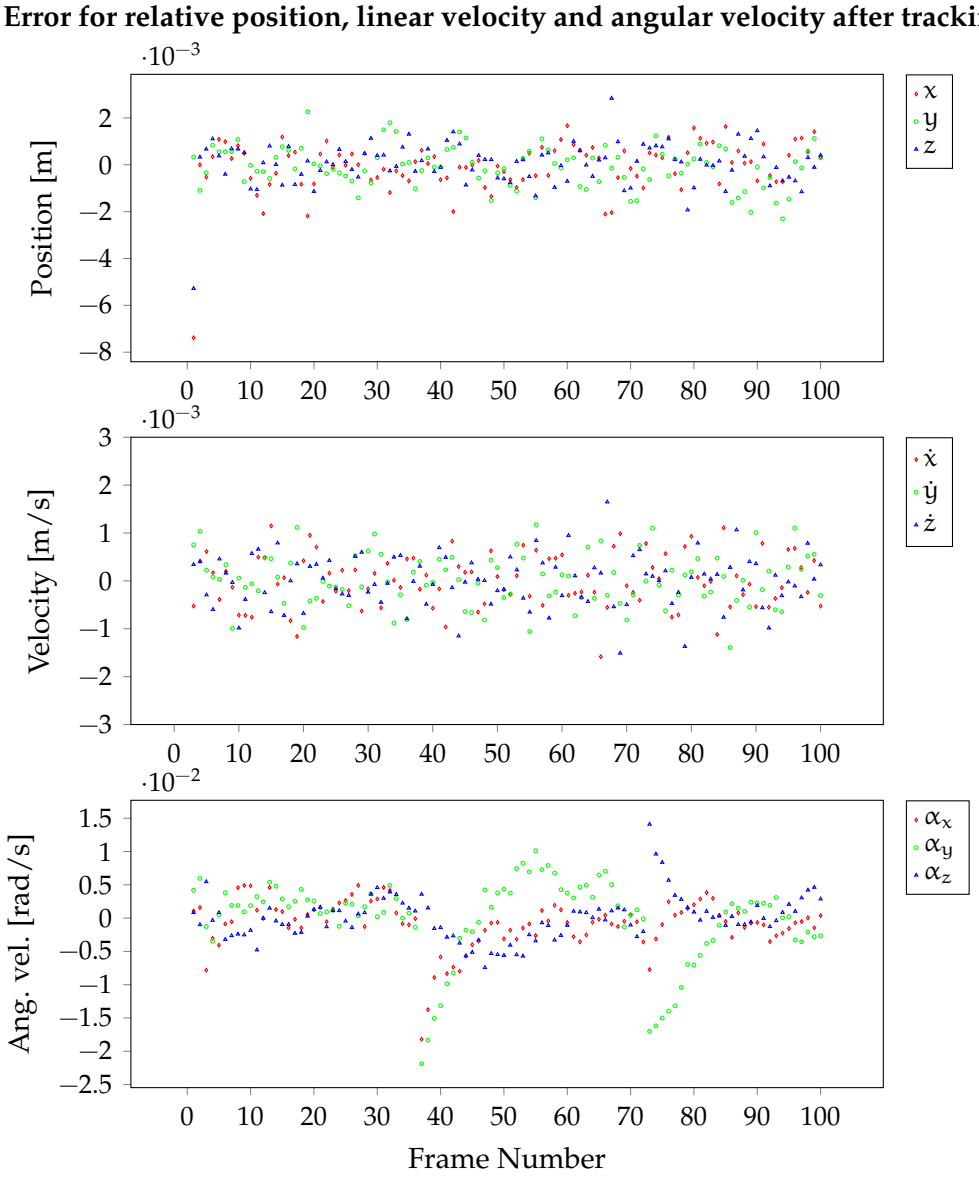


Figure 5.10. Difference between the EKF-estimated and the true relative rocket position, velocity and angular rate.

gure 5.12, which shows the difference between the true and estimated values for the Euler angles associated to the upper stage's attitude. The roll (ϕ) and pitch (θ) angles were estimated with an accuracy of 3°, whereas the yaw angle (ψ) is less accurate. The latter was due to the characteristics of the simulated tumbling motion and the tracking procedure employed: the yaw angle estimation is the most affected by the registration errors such the one visualized in Figure 5.8.

Then again, since the fairing was selected as the distinctive point on the rocket surface to be mapped by inclusion of its sensor-frame coordinates in the state vector, the yaw angle estimation benefits the most from re-initialization: the steep correction at each full turn is evident and solves the problem of unbound drifting that a tracking-only approach would manifest. Also, this steep correction explains the spike in the angular velocity (see Figure 5.9) estimation, which obviously reacts proportionally to the amount of the yaw angle correction that follows the re-initialization.

5.2 Real Data

In order to validate the algorithms in real world, the construction of a scaled model of the Cosmos-3M upper stage was carried. The scale of 1:10 was determined using different cylinder models, which were built with cardboard with the same dimension relationships but different scales. Figure 5.13 shows the diverse models employed to determine the adequate size for the algorithms assessment. Having all the models, diverse point clouds were taken with the PMD-Technologies CamCube 3.0 (see Section 2.1.1 and Figure 2.2) at different cylinder positions. Later on, a global descriptor for each of them was estimated in order to analyze if the point clouds shared any kind of relationship. Thus, the use of a scaled model could be effective for the algorithm validation, without having a real size rocket body. The geometrical similitude based on global descriptors can be observed in Figure 5.14.

Also the level of noise in the data was analyzed in order to have the most clean point cloud, but without having a very big mock-up. The latter was a constrain of the maximum print volume of the 3D printer (MakerBot, 2014). While it is possible to segment a body, print all the pieces, and assembly them at the end, the resultant model does not have a smooth surface compared to a model that is constructed in just one piece. Taking into account all those restrictions, as well as the scale ambiguity (see Figure 2.6), the most suitable option was a model of 1:10 in scale

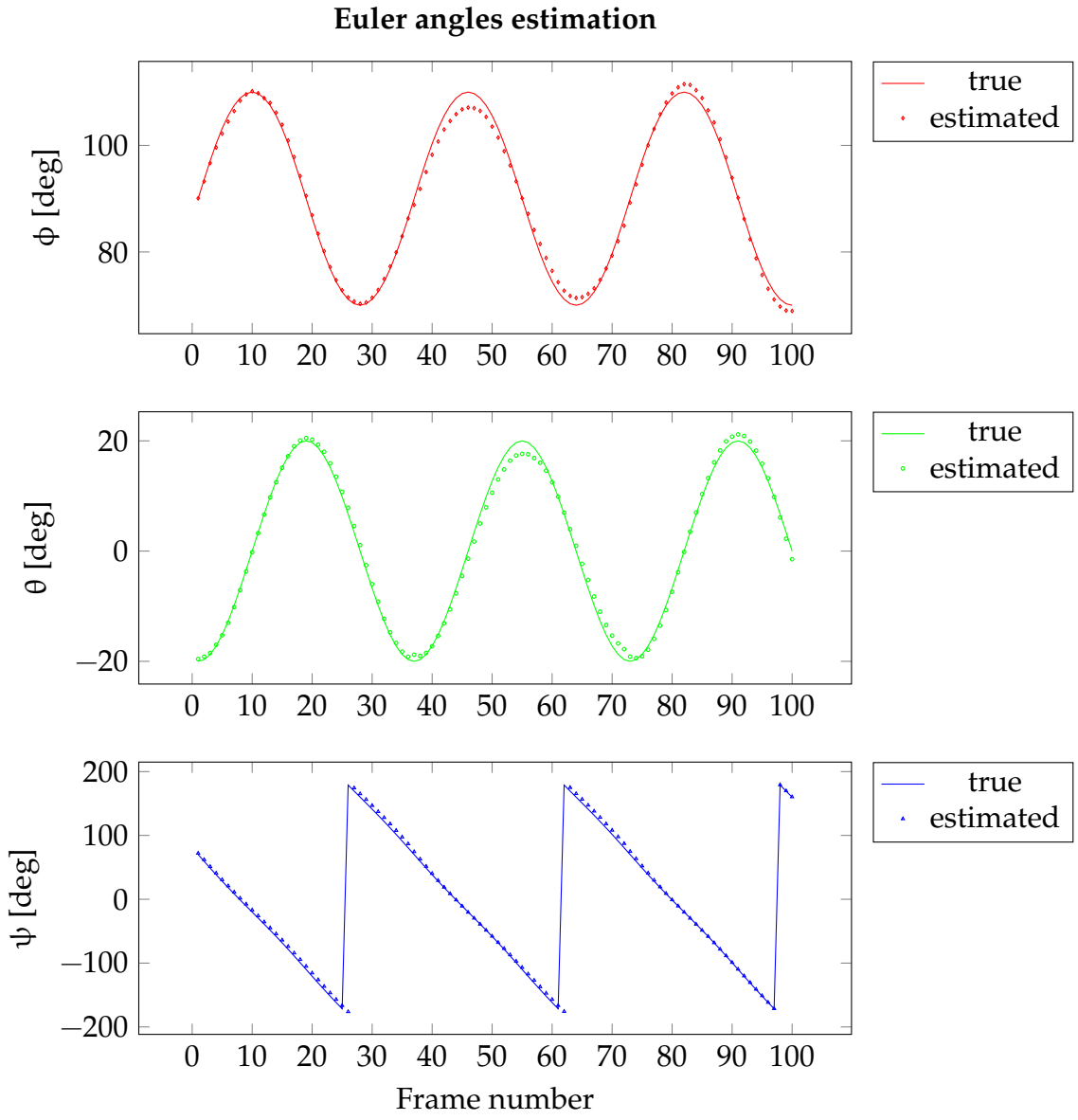


Figure 5.11. Attitude estimate output by the EKF, given in Euler angles, compared to the true values.

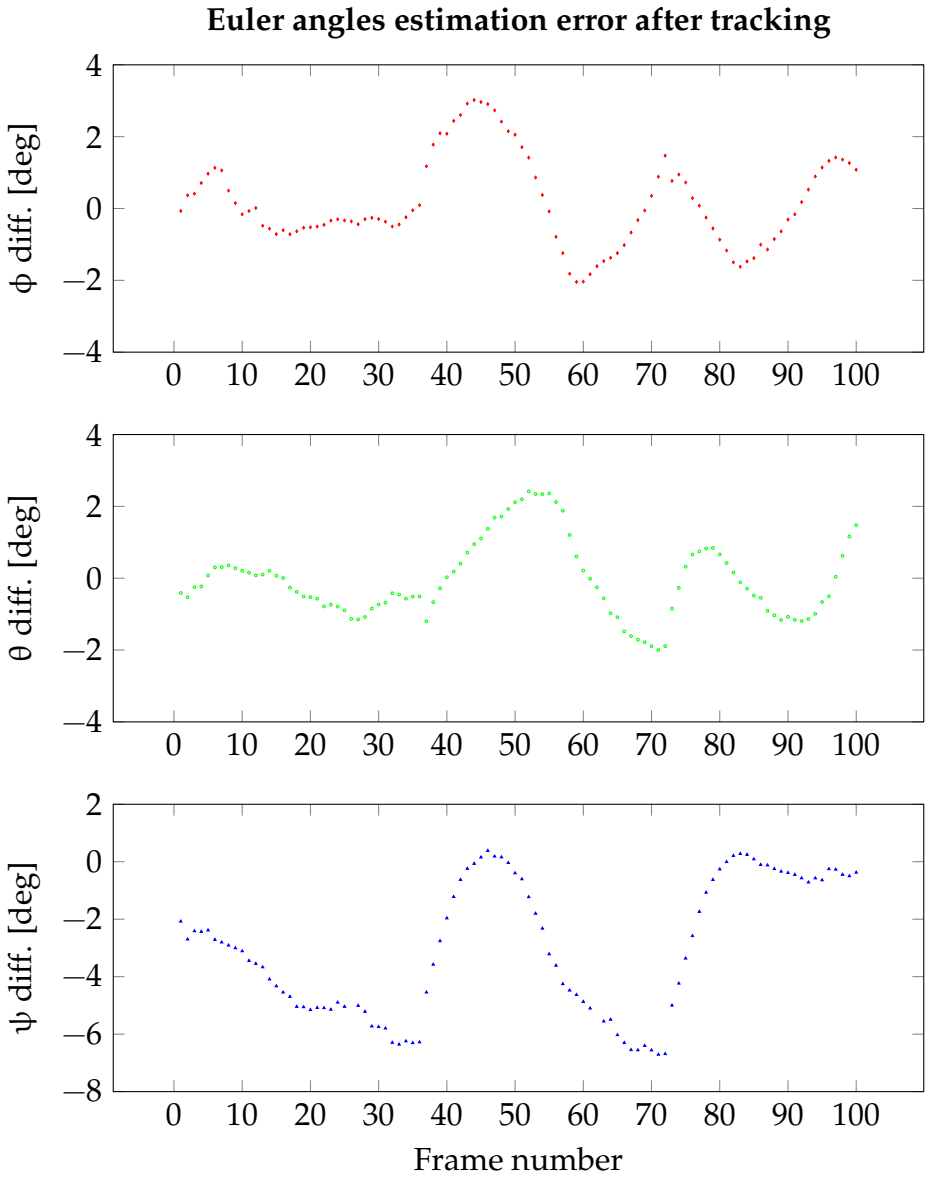


Figure 5.12. Difference between the EKF-estimated and the true relative rocket orientation, given in Euler angles.

Figure 5.13. Cylinder mock-ups for scale assessment.

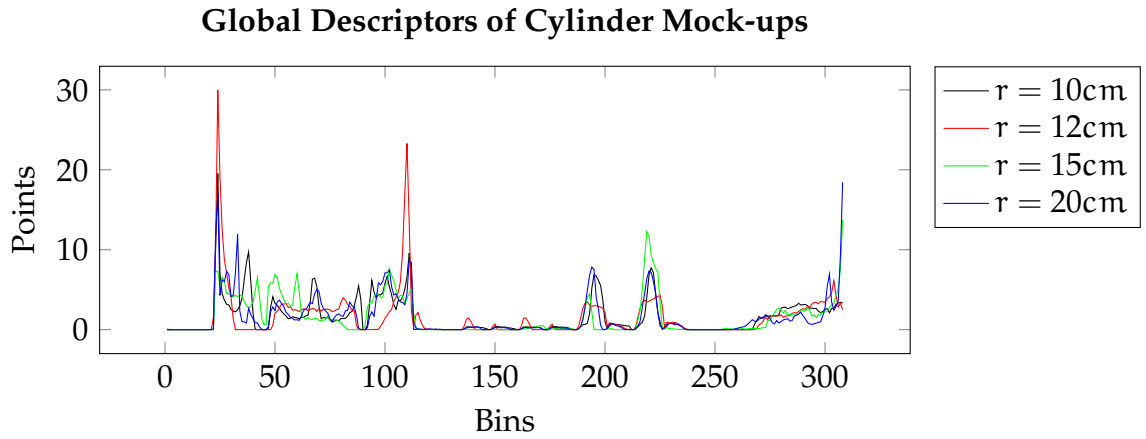


Figure 5.14. OUR-CVFH descriptor comparison for cardboard cylinder mock-ups. Cylinder models with smallest radius were not taken into account because of the high level of noise in the data, i.e. saturation in sensor, hence it was not possible to fit the RANSAC cylindrical model.

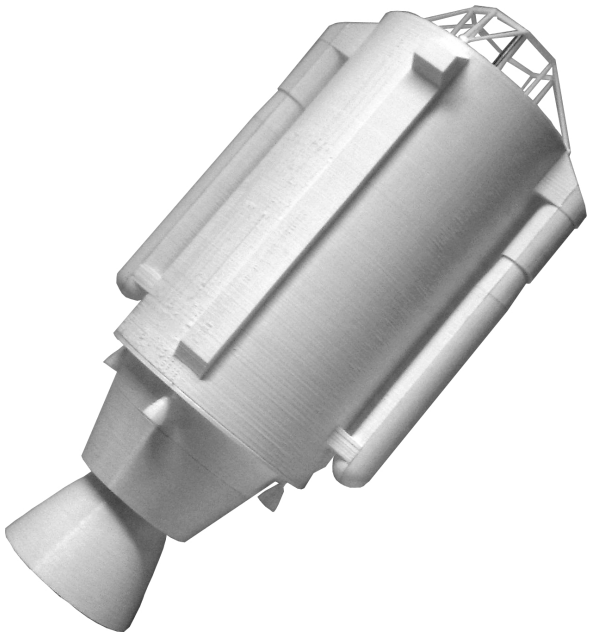


Figure 5.15. Scaled, printed model of the Cosmos-3M upper stage.

with respect to the assumed real rocket body dimensions, i.e. the cylinder of radius $r = 12cm$. Figure 5.15 shows the outcome of the 3D printed rocket body, as well as Figure 5.16 presents a point cloud sample obtained from the printed target.

In order to simulate a tumbling behavior of the rocket body, the latter had a fixed orientation, and put onto a rotating table to provide an angular velocity around the vertical axis. This was done is angle steps of 5°, and the ToF camera recorded each pose in a "stop-and-go" manner. Due to technical challenges, it was not possible to record a full-360° turn, because the support frame, which held the rocket body model in the given orientation, blocked the model from the sensor's point of view. Figure 5.17 shows a render of how the data was recorded.

5.2.1 Pose initialization based on Body Geometry

After adaptation of the algorithms to the new target size, the procedure remained the same as for the simulated test, but with different initial conditions, indicated in Table 5.2. However, contrary to the synthetic data analysis, only the pose initialization based on the body geometry was assessed. The other method, i.e. that based on global descriptors, was not taken into account due to the great number of issues this procedure has, like the training data set generation, or the rejection of false positives

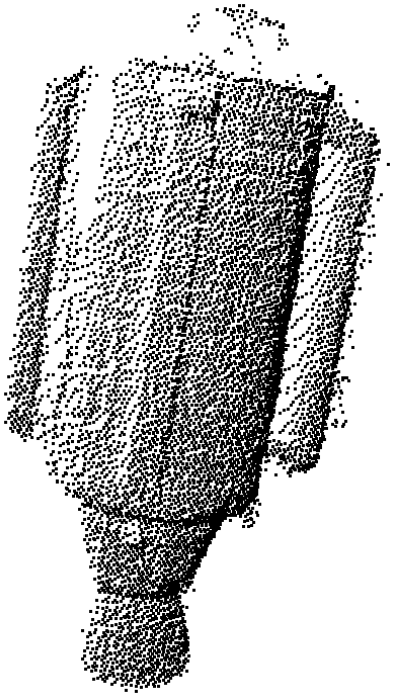


Figure 5.16. Point cloud obtained from the printed model using the CamCube 3.0. The noise in the data is noticeable.

Figure 5.17. Render of the real point cloud data acquirement.

Table 5.2. Real Data Simulation - Body Initial Parameters

Parameter	Value
Roll angle (about sensor's x-axis)	$-90°$
Pitch angle (about sensor's y-axis)	$-112.5°$
Yaw angle (about sensor's z-axis)	$-40°$
Rotation axis in sensor reference frame	$[0, 1, 0]$
Angular velocity	$5°/s$
Sensor acquisition frequency	1Hz
Number of frames	45

in the closest neighbor matching, among others. Figure 5.18 shows the result of the initialization procedure for a specific pose. For the given range of samples, the result of the pose initialization is shown in Figure 5.19, having as the true value the nominal orientation of the body, taken from the CAD model in Blender. Apparently, the noise reduction preprocessing can produce loss of significant data that could belong to the surface of the target instead of being spurious points floating anywhere.

Nevertheless, the initialization was possible to perform well in general terms, compared to the simulated counterpart (see Figure 5.4). Figure 5.20 indicates the error for each Euler angle is around $4°$, showing that the noise in data is a determinant factor for the pose initialization's lack of accuracy. As a comment, the plot at the bottom of Figure 5.20 does not show the error difference for the last 9 frames. These data was excluded due to the big difference between the true and estimated values for the yaw angle, when the fairing is not observable by the sensor in its FOV.

5.2.2 Pose Tracking

As its counterpart with the simulated data, the tracking phase for the printed rocket body has the rigid transformation estimation plus the state vector determination using the point cloud registration pipeline and the EKF, respectively. Figure 5.21 shows the registration error, where the latter is not exceeding more than 6mm when both coarse and fine alignments have been performed in order to extract the rigid rotation matrix. Compared to the simulation, the error value is approximately reduced by one order of magnitude, giving robustness to the procedure. The latter refers to the reduction of dimensions of the rocket body for proper manufacture and real data analysis.

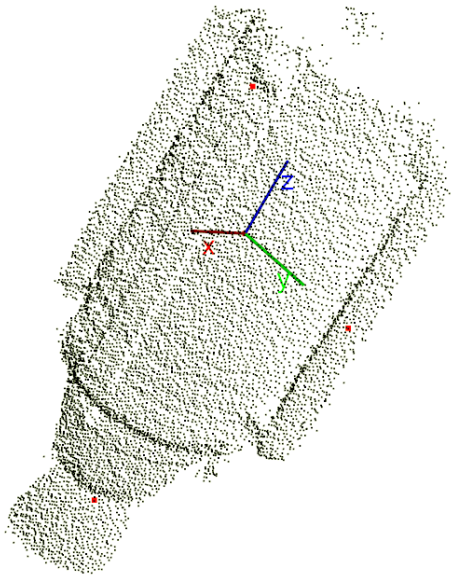


Figure 5.18. Outcome of the initialization process based on body geometry on the printed model.

After the initialization of the rocket body printed model, the estimation of the state vector is performed in the same manner as that used for the synthetic data, described in Section 5.1.3. Thus, the initial values for the position, linear velocity, orientation and angular velocity of the printed model are the input data for the EKF. In this case the tracking filter converged rapidly towards the dynamic parameters, as can be seen in Figure 5.22. The error of the estimation compared with the ground truth values are shown in Figure 5.23.

However, this convergence cannot be seen with respect to the Euler angles that described the orientation of the printed rocket body. These results are shown in Figure 5.24 in Euler angles. Similarly to the results shown in Section 5.1.3, the truth in the tracking procedure of the rocket orientation is shown in Figure 5.25. As expected, the accuracy has been reduced because of the noise in the data. The roll (ϕ) and pitch (θ) angles were estimated with an accuracy of 10° in average (sometimes more), whereas the yaw angle (ψ) could achieve a variation up to 20°. This confirms the effects of the body's symmetry, as observed with the results obtained for the simulated scenario (see Figure 5.12) This loose on the estimation makes a possible real rendezvous on orbit to be an arduous task. However, it is important to take into account the additional artifacts in the point clouds used in this research could be

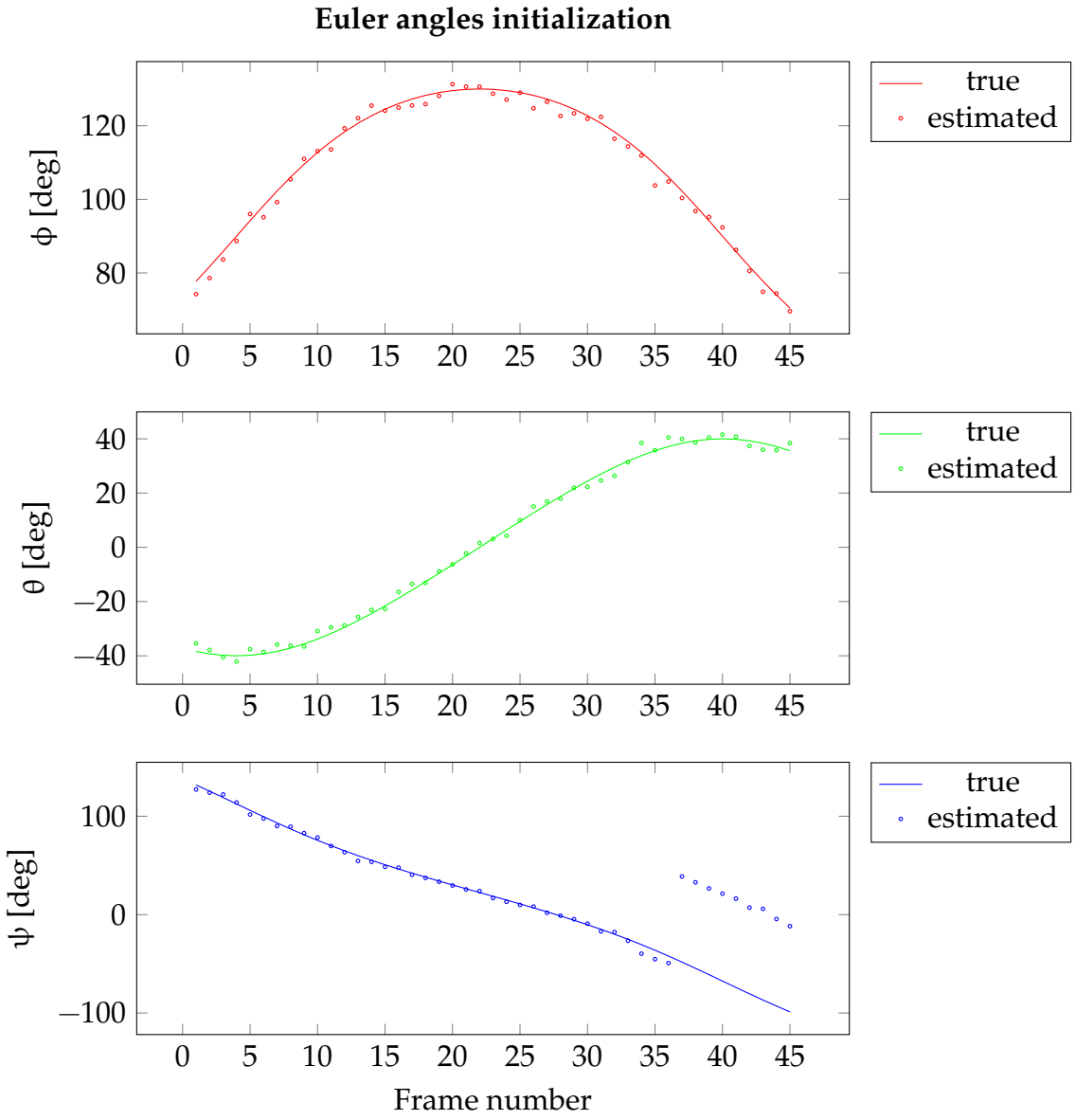


Figure 5.19. Output of the initialization routine for estimating the scaled-rocket orientation. Both estimated and nominal (true) Euler angles are given. The nine incorrect initializations of the rocket yaw angle are due to the lack of view of the fairing from the sensor, similar to the situation of the simulated point cloud.

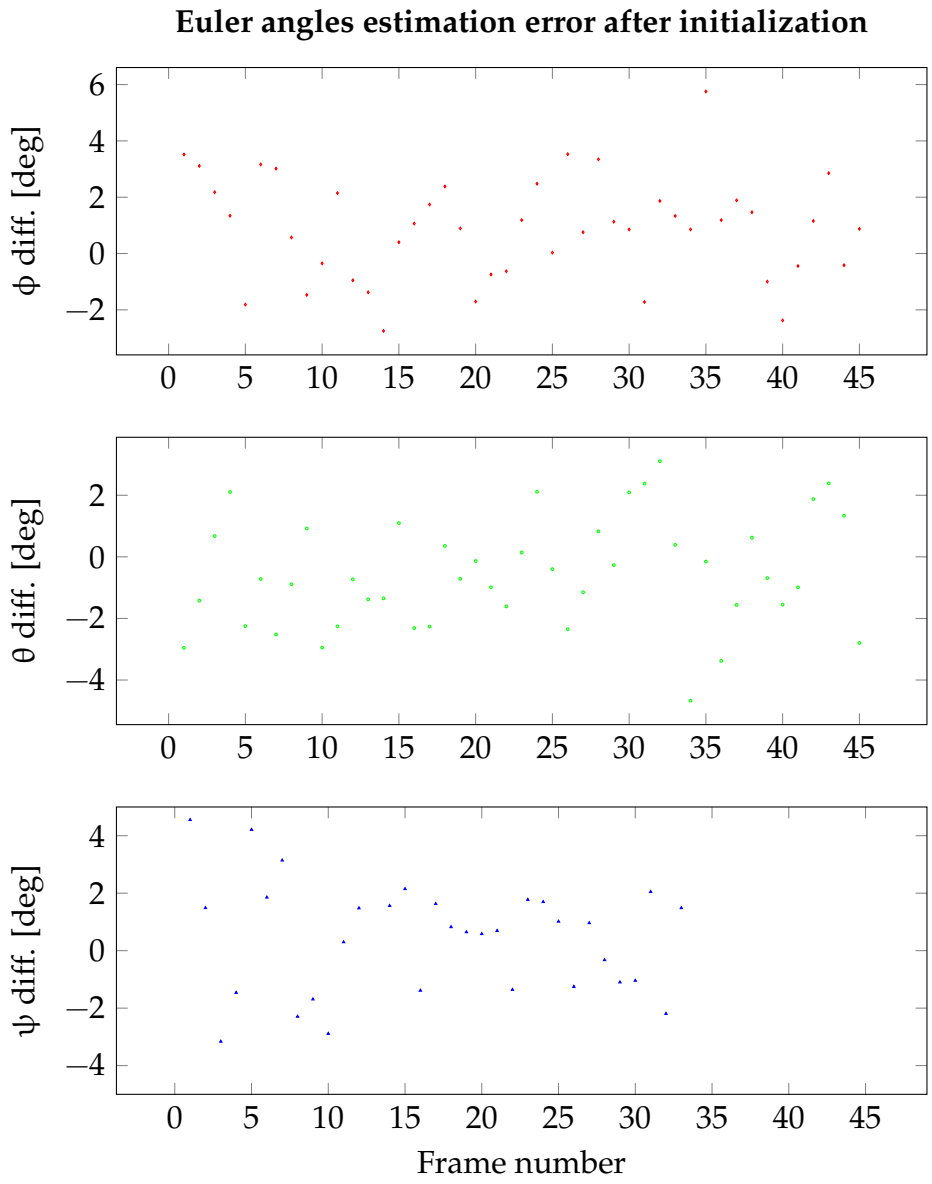


Figure 5.20. Difference between the estimated and true relative orientations for the printed rocket body, given in Euler angles, following the initialization procedure.

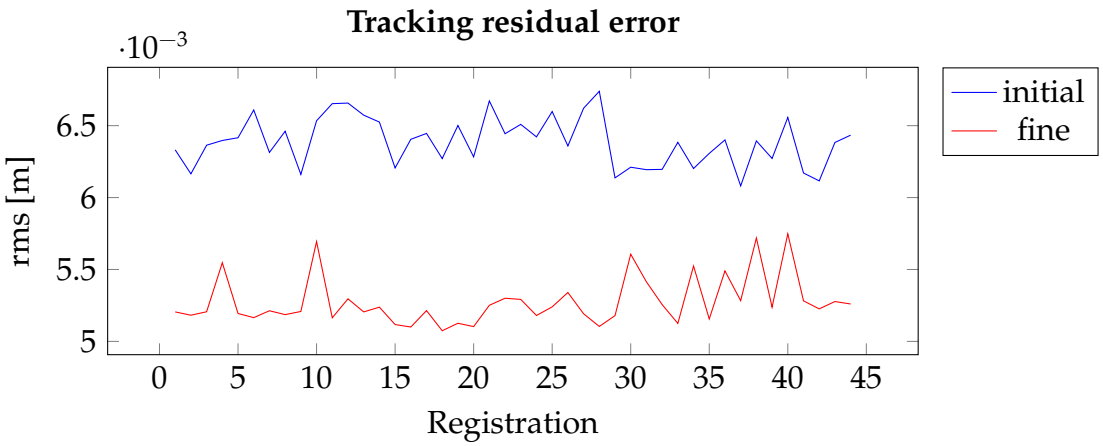


Figure 5.21. Residual error for the real data alignment of each pair of consecutive frames. The blue curve shows the coarse alignment using SVD. The red curve shows the fine alignment using ICP, after the coarse one.

caused to different conditions, i.e. the material and size of the rocket body used, or the scene illumination, that might not be present in real operation.

5.3 Summary

This chapter presented a condensed view for the validation tests of the pose initialization and tracking of a rocket body. Using both synthetic point clouds and real ToF camera data, the algorithms could obtain plausible results based on the uncertainty given by point clouds with no additional clues apart of their spatial distribution. The synthetic data simulation demonstrated the drawbacks of the pose initialization using two different methods. For instance, the global descriptors requires a lot more of data preprocessing for creating the training data list, as well as the necessity of hypothesis validation for the closest matches found. In order to reduce the number of false positives, more data must be submitted to the training list, with a cost in the memory size of the system, and the rise of the computational burden for the brute-force descriptor comparison. Contrary to the body geometry-based initialization, where the pose estimation did not require any matching method, but dependent on fitting in geometrical primitives. Also, the absence of vital structures in the sensor FOV produced a bad estimation.

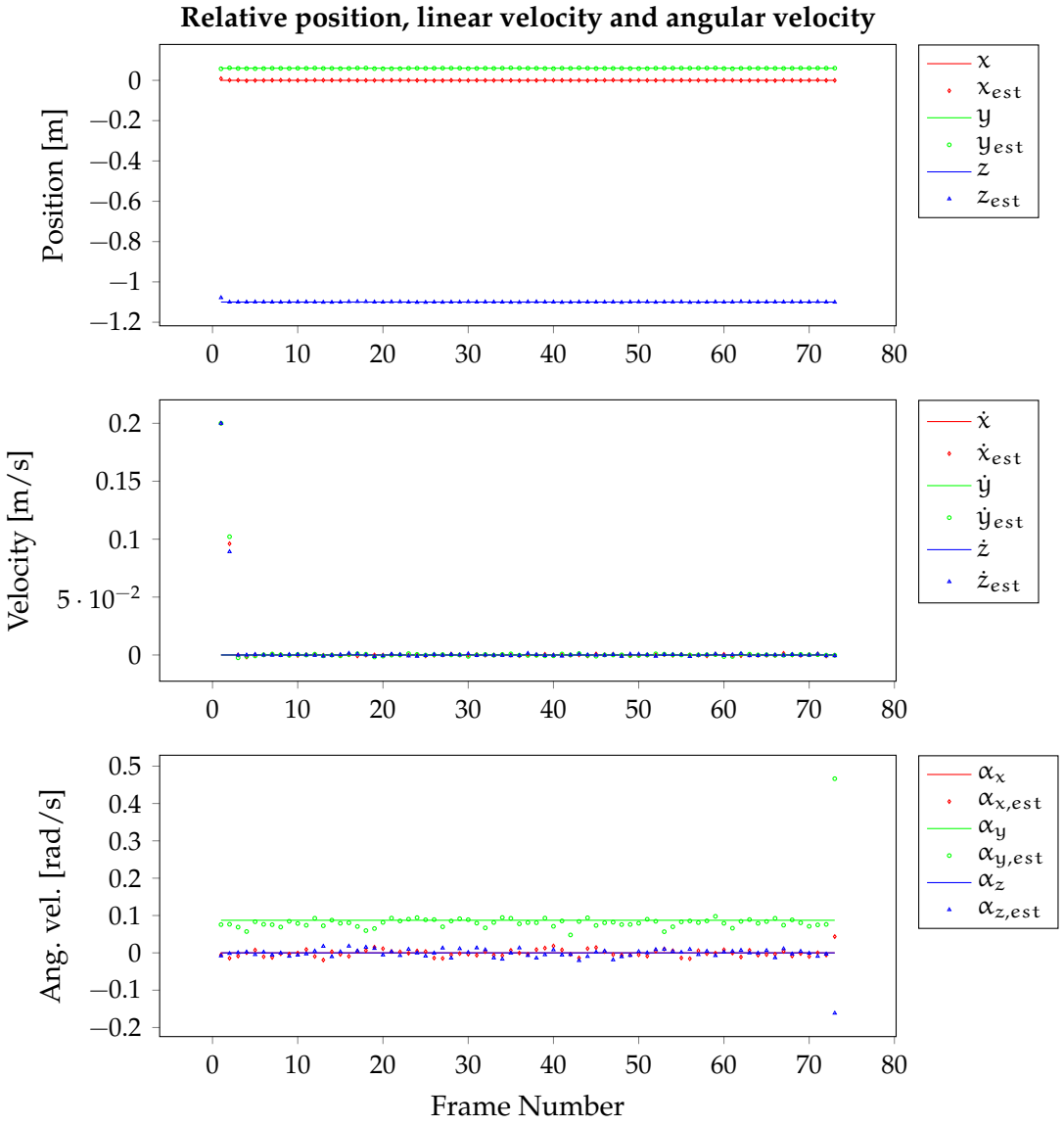


Figure 5.22. Comparison of the true and estimated relative position, linear velocity, and angular velocity output by the EKF for real sensor data.

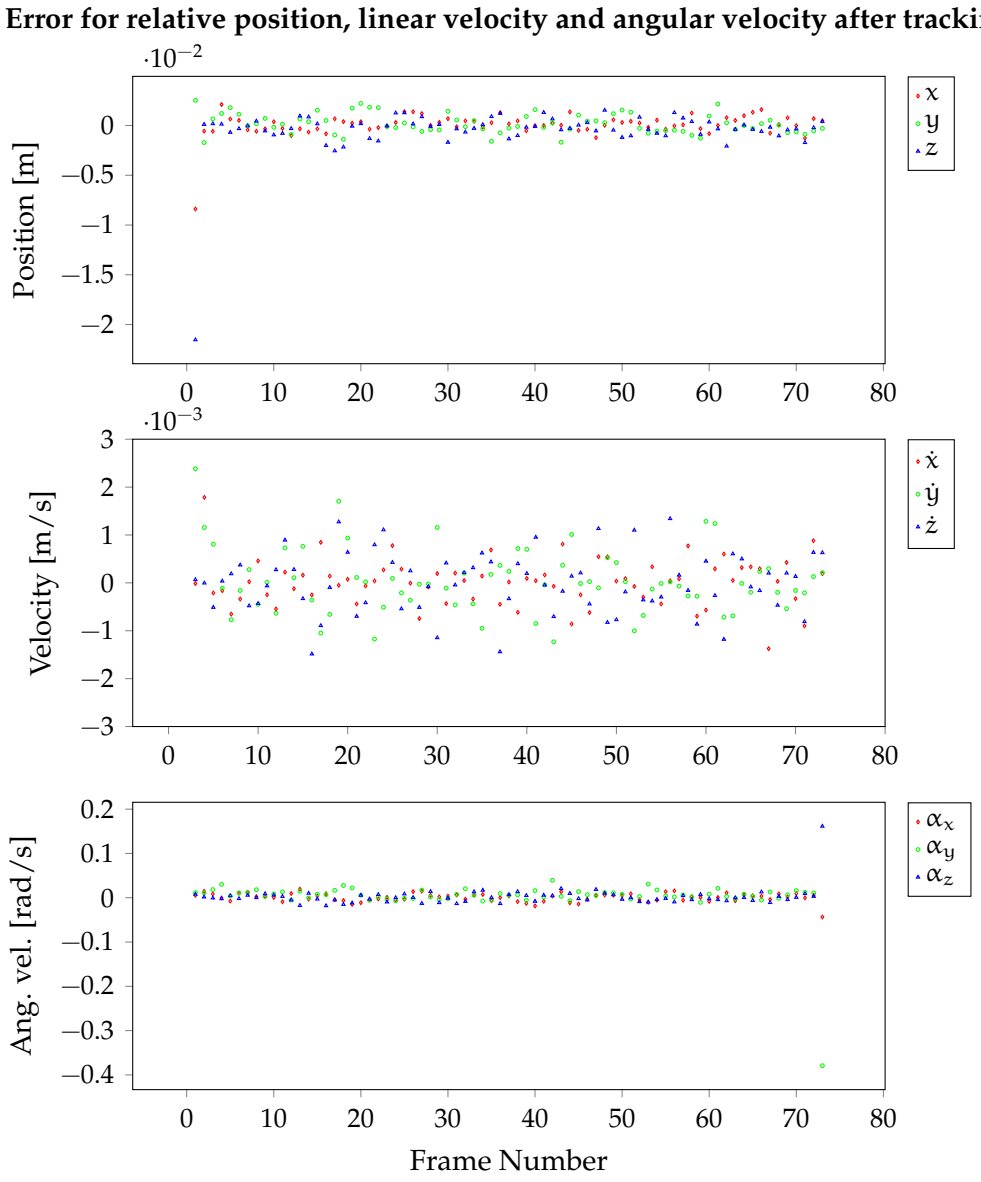


Figure 5.23. Difference between the EKF-estimated and the true relative rocket position, velocity and angular rate for real sensor data.

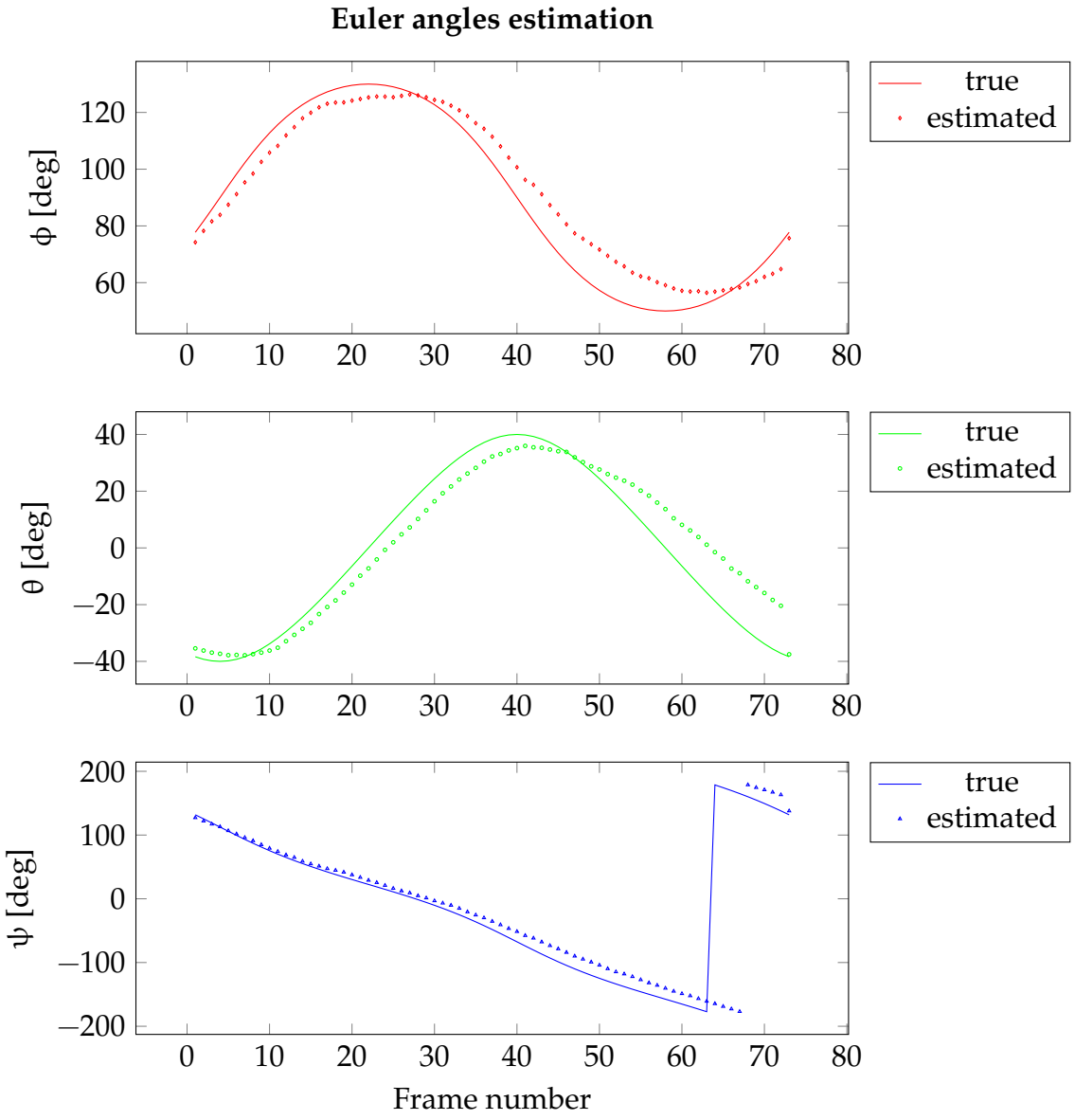


Figure 5.24. Orientation estimate output by the EKF using real sensor data, given in Euler angles, compared to the true values.

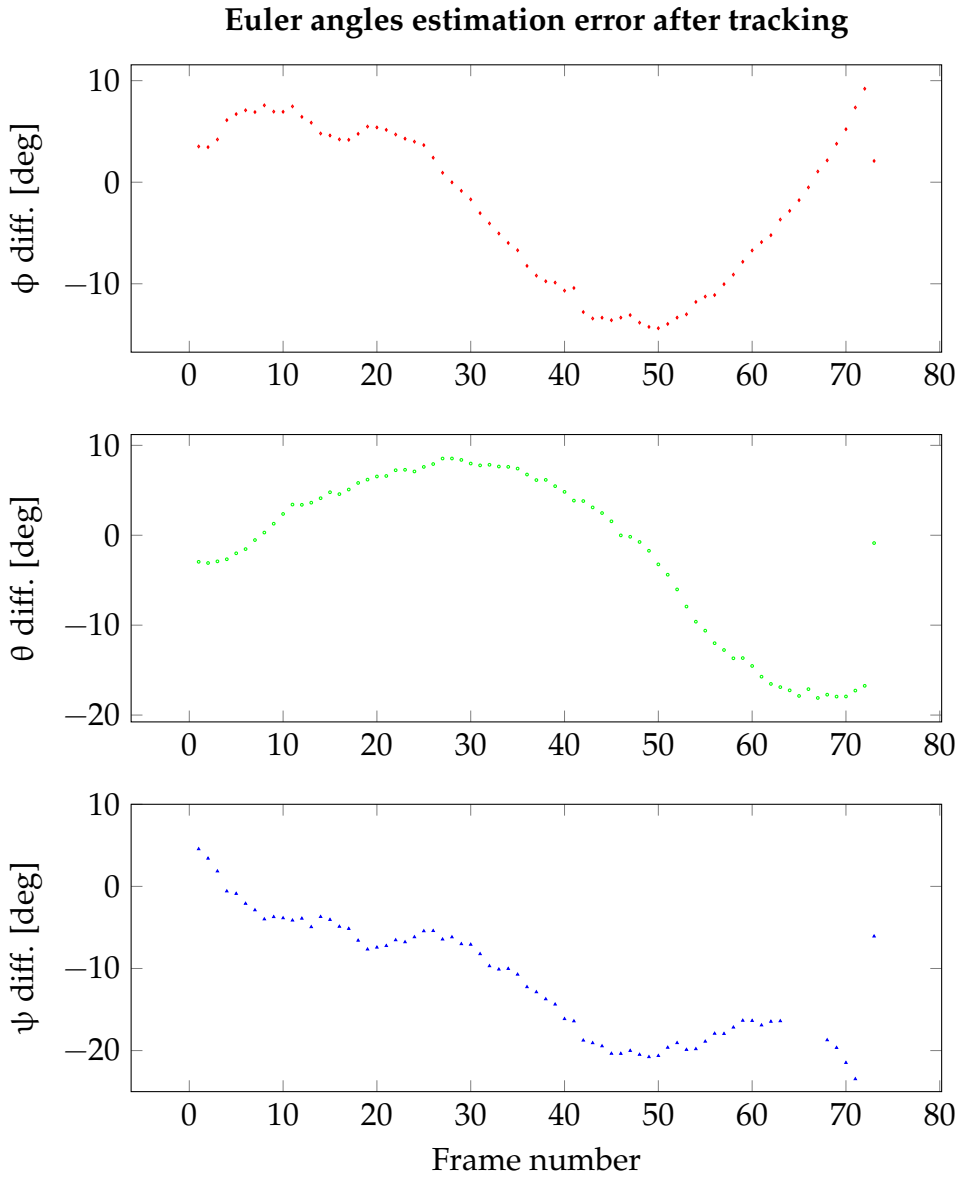


Figure 5.25. Difference between the EKF-estimated and the true relative printed rocket orientation, given in Euler angles.

Additionally, the pose tracking demonstrated to be steadier for synthetic data than for real data. The latter was influenced on the control level that a simulation may have in comparison to real data collection. The EKF results for the simulation were confident enough to consider the position and orientation values obtained as useful for an eventual target capture. On the other hand, the outcome of the real data requires additional enhancements in the observation methods, in order to produce better results in the state vector estimation. As a closing remark, it would be hard and unfair to compare the accuracy between the synthetic and real data evaluations, because the noise for each case affected in different way the diverse steps in the whole estimation pipeline. However, the foundation for the real data test was the algorithm developed using the synthetic model of the upper stage.

6 | Conclusion

This thesis has pointed towards the possible solution of the reduction in the space debris population. ADR is a natural response from space agencies and universities in order to solve this issue: this work intends to offer a solution for detecting a chosen target, and estimating its relative position and orientation with respect to a chaser satellite. This navigation approach is based on the use of ToF cameras, whose operation principle is based on the same used by the flash lidar, but with lower power requirements, reduced mass, and lack of mechanical devices.

Although the space debris have been cataloged in different groups, e.g. based on their size or on their operation purpose, the rocket bodies were selected as the main target for possible active removal. The latter is due to the probability they have in order to produce additional debris, by possible impact to other objects, sudden explosion of remained fuel, or survival of an uncontrolled reentry.

The main objective of the visual navigation algorithm proposed in this research was to identify a non-cooperative, free-tumbling rocket body on orbit using a ToF camera. Then, the routine performed the identification of the translational and rotational states of the target, in order to have an initial pose estimation. Subsequently, the algorithm followed the motion of the object, tracking it frame to frame with the inclusion of point clouds procedures, and an extended Kalman filter in order to have a more accurately estimation the upper stage's pose.

Despite of the diverse non-cooperative objects that may be labeled as target for ADR missions, this research was only focused on the disposal of rocket bodies. There are diverse kind of space debris, whose shape is not cylindrical, but prismatic, or even irregular. For those targets, a different approach for pose determination is required. In other words, there is no one solution in order to cover all the different types of space debris. In addition to this, the proposed method only takes into

account the contribution of a single sensor for performing the pose estimation task, based on its advantages in terms of mass and power consumption. However, sensors with better illumination capabilities (i.e. a lidar) in conjunction with multi-spectral passive cameras, can give more robustness to the relative navigation system. Nevertheless, this enhanced procedure will increase the complexity of the system, in terms of both hardware and software.

All in all, this chapter is a summary of the contributions made, based on the results obtained thanks to the simulated point clouds and real data taken with a ToF camera on a scaled rocket body model. Additionally, it addresses diverse proposals of future research that can possibly enhance the scope of this work.

6.1 Contributions

The outcome of this research work was the development of a pipeline to detect a designated target, to obtain the 6DOF pose at a given epoch, and to track the evolution in time of the object's motion, using solely point clouds as input data. To achieve those goals, different approaches were employed to find the appropriate solution for each one of those phases.

The recognition and initialization procedures are bound, because the latter cannot be executed without the former. Taking advantage of the general shape of rocket launchers, whose main structural body is a cylinder, the algorithm evaluates the obtained point cloud, fitting a cylindrical model to the data by means of the RANSAC method. The effectiveness of this process is only assured if a cylinder model of certain radius fits in the obtained data. If this is not possible, the algorithm does not go further until it matches a suitable cylindrical model.

Then, the detection of additional structural parts, like the nozzle, fairings or external tanks, is crucial for confirming if the object under observation is the target chosen for the ADR mission. The recognition of the target's appendages is also accomplished assessing if the extension of the normal vectors of other structures also coincide with the main body's axis of symmetry –for the nozzle–, as well as using the RANSAC method for distinguishing the external tanks and the fairings, if any. Also, the relative position of those structural elements helps to resolve the

paradigm of finding the position and orientation of the rocket body, relative to the ToF sensor.

As a consequence, the routine can start the motion tracking phase. For this purpose, the ICP registration algorithm carries this workload to achieve such objective. The ICP algorithm evaluates consecutive point clouds in order to determine the rigid transformation between them, finding the rotation matrix and the translation vector between frames. However, the use of an estimation filter is required, because the sensor does not provide exact measurements, making the registration process not as accurate as required. For that reason, the use of an EKF is important to take those measurements and have a better estimation of the target's state vector. Here, the inclusion of a loop closure point is necessary in order to decrease the drift inherent to the EKF operation, making the pose estimation a bit preciser.

Finally, the framework was validated though experiments using both synthetic data from the CAD model of the rocket's upper stage, and real data obtained from a PMD-Technologies CamCube 3.0 ToF camera. The latter has a 3D-printed model as the target, in a 1:10 scale. Although the construction material of the model differs from the real rocket body, the obtained data also contained noise, that could not be achieved by simulation means. Those additional sources of error were relevant for testing the robustness of the pose estimation algorithm.

6.2 Future Research

After assessing the outcome of the routines presented, other topics have risen in order to enhance the performance of the algorithm: to make the latter more efficient, more autonomous and more accurate at the target's state estimation.

6.2.1 Fusion with 2D Images

The usage of range sensors allows to obtain a 3D representation of the target's surface in a direct manner. Additionally, the obtained range map grants a first estimation of the relative distance between the target and the chaser satellite, without additional calculations. Furthermore, the ToF camera has more robustness in terms of illumination conditions, and they can almost operate in both light and dark

conditions. However, the strong dependency on the point data distribution makes the registration procedure to produce misleading results. The latter is a consequence of bad initialization on the rigid transformation between the clouds at the coarse alignment.

In order to make the registration less prone to errors, the association with 2D images could enhance the accuracy of the initial alignment. Using image feature descriptors like the Scale Invariant Feature Transform (SIFT) (Lowe, 1999), the routine can detect specific keypoints on the image, whose signature is robust against translation, scaling, and rotation. When those keypoints are detected, they can be fussed with the 3D surface's point cloud, giving an additional factor for correspondence between frames. Hence, it is possible to reduce the ambiguity in the association of 3D descriptors related to symmetrical surfaces.

6.2.2 Online Model Generation

This work assumed the knowledge of the target geometry before any capture attempt. Unfortunately, there is complete uncertainty about the real condition of a rocket body or any inoperative satellite. Because of that, it is crucial to know in advance the real conditions of the target, e.g. if its integrity is still maintained, or if its surface had been affected by a collision, or eroded due to temperature changes. Thus, a possible target model reconstruction would enhance the overall process of target identification, including the pose initialization and further tracking.

Here, the fusion of 2D and 3D data exposed in Section 6.2.1 is fundamental for recreating an online model. For instance, the use of 2D features in association with range data has been investigated in past (Padial et al., 2012), in order to create an ad hoc 3D model. Here, it would be useful to have the point clouds registration pipelines, using with the 2D feature data in order to have a better correspondence association, increasing the accuracy of the routine. The object model would be finished when the algorithm determines if the features observed in a later epoch has been already seen in the past. This would determine the loop closure of an expected turning behavior of the target. After having a complete model, the determination of the body fixed reference frame should be defined based on the mission's requirements. However, this new procedure may carry additional computational burden,

that have to be taken into account for the 2D feature detection and tracking, as well as the determination of the loop closure, which both would be memory intensive.

6.2.3 Real-time Implementation

The simulations carried were tested for offline operation, i.e. the algorithm was not designed to operate in real time. All point clouds, either synthetic or real data obtained from the ToF camera, were taken in advance and stored for further operation. If the 2D feature descriptors or the ad hoc model generation are included, the rise in the number of operations is noticeable, as well as the signatures comparison procedure for both 2D and 3D loop closure.

Additionally, online pose detection and estimation requires that the data coming from the sensors is analyzed just immediately after acquisition, having the proper balance between accuracy of the descriptors' evaluation and the speed of the involved calculations. Hence, due to the high load of computation for 2D and 3D computer vision, a topic for investigation would be parallel computing. The latter is currently achieved by Graphical Processor Units (GPU) in most of the robotics research projects, but apparently with no presence in space robotics. Having parallelization in the computer vision pipelines might produce a better performance in the many computations the system has to run frame to frame.

6.2.4 Non-linear Estimation Filters

Although the EKF performed well in the tracking phase of the algorithm, its formulation and respective tuning made it a bit tedious, because of the Jacobian matrix calculation. For this reason, an additional research could be pointing towards the use of another alternatives of non-linear estimation filters, like the Unscented Kalman Filter (UKF) or the Particles Filter (PF).

Those previously mentioned examples might accomplish the target's state estimation in a better way, more accurate and faster than the EKF used in this research. The avoidance of calculating derivatives of non-linear functions gives great advantages at the filter design phase. Otherwise, they can be also eager in terms of computational resources. An exhaustive trade-off must be made in order to assess if the

inclusion of a different estimation filter may have a real advantage in comparison to the filter used in this work.

6.2.5 Dual Control with Enhanced Target Models

For both synthetic and real simulations, the chaser satellite was assumed to be at a fixed position with respect to the target, maintaining a tumbling motion around the center of mass. For a expected mission of ADR, the chaser satellite may require to inspect the target from different locations, depending on the motion that the latter may have. Those approach maneuvers should also be taken into account for pose estimation, and for the model construction, if it will be included. This approach of combining the estimation problem with the control one is called "dual control" (Kim and Rock, 2006). This contributes to have small errors in both estimation and control issues. This concept can also be applied for the trajectory planning when the agents would chase the target. Maybe the chosen path would not be the shorter one, but it could be more efficient, reducing the estimation uncertainty.

On the other hand, the enhancement in both synthetic and mock-up models will allow a better data reproduction for the algorithm application, with the appropriate illumination conditions in orbit. Regarding the CAD model, it may be difficult to simulate the exact reflection properties of the rocket body, but it can provide a better understanding of how the images and point clouds could behave. If the chosen target has Multi-Layer Insulation (MLI), e.g. OOS or a defunct satellite for ADR, it may be more difficult to simulate the appropriate reflection that this material has. With respect to the mock-up, the material used for its construction is not the same as that used for real rocket body manufacture. Additionally, the type of coating used on the launchers has specific properties, and this changes among primer manufacturers. Thus, the proposed idea is to have a similar mock-up, in size –if possible– and in manufacture material. Regarding the finishing of the target's surface, it could be quite impossible to match the same type of paint and the real conditions of the painting with respect to a real object in orbit, which has been under strong environmental conditions, provoking erosion and/or delamination of the coating. But the results might be closer to those expected to have in orbit conditions.

Bibliography

Aghili, F. (2010). Automated Rendezvous & Docking (AR&D) without impact using a reliable 3D vision system. In *AIAA Guidance, Navigation, and Control Conference*, pages 1–11.

Aghili, F. and Parsa, K. (2009). Motion and parameter estimation of space objects using laser-vision data. *Journal of Guidance, Control, and Dynamics*, 32(2):538–550.

Aldoma, A., Marton, Z. C., Tombari, F., Wohlkinger, W., Potthast, C., Zeisl, B., Rusu, R., Gedikli, S., and Vincze, M. (2012a). Tutorial: Point cloud library: Three-dimensional object recognition and 6 DoF pose estimation. *IEEE Robotics and Automation Magazine*, 19(3):80–91.

Aldoma, A., Tombari, F., Rusu, R. B., and Vincze, M. (2012b). OUR-CVFH – oriented, unique and repeatable clustered viewpoint feature histogram for object recognition and 6DOF pose estimation. In *Pattern Recognition. DAGM/OAGM 2012. Lecture Notes in Computer Science.*, pages 113–122.

Aldoma, A., Vincze, M., Blodow, N., Gossow, D., Gedikli, S., Rusu, R. B., and Bradski, G. (2011). CAD-model recognition and 6DOF pose estimation using 3D cues. In *IEEE International Conference on Computer Vision*, pages 585–592.

Arun, K. S., Huang, T. S., and Blostein, S. D. (1987). Least-Squares fitting of two 3-D point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):698–700.

Besl, P. (1988). Active, optical range imaging sensors. *Machine vision and applications*, 1:127–152.

Besl, P. J. and McKay, N. D. (1992). A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256.

Biesbroek, R., Innocenti, L., Wolahan, A., and Morales Serrano, S. (2017). e.Deorbit – ESA's Active Debris Removal Mission. In *7th European Conference on Space Debris*, pages 1–10.

Boge, T., Benninghoff, H., and Tzschichholz, T. (2013). Visual navigation for On-Orbit Servicing missions. In *5th International Conference on Spacecraft Formation Flying Missions and Technologies*, pages 1–11.

Bonin-Font, F., Ortiz, A., and Oliver, G. (2008). Visual navigation for mobile robots: a survey. *Journal of Intelligent and Robotic Systems*, 53:263–296.

Bonnal, C., Ruault, J. M., and Desjean, M. C. (2013). Active debris removal: Recent progress and current trends. *Acta Astronautica*, 85:51–60.

Christian, J. A. and Cryan, S. (2013). A survey of LIDAR technology and its use in spacecraft relative navigation. In *AIAA Guidance, Navigation, and Control Conference*, pages 1–7.

Christian, J. A., Patangan, M., Hinkel, H., Chevray, K., and Brazzel, J. (2012). Comparison of Orion vision navigation sensor performance from STS-134 and the Space Operations Simulation Center. In *AIAA Guidance, Navigation, and Control Conference*, pages 1–18.

Crassidis, J. L., Markley, F. L., and Cheng, Y. (2007). Survey of nonlinear attitude estimation methods. *Journal of Guidance, Control, and Dynamics*, 30(1):12–28.

D'Amico, S., Ardaens, J.-S., Gaias, G., Schlepp, B., Benninghoff, H., Tzschichholz, T., Karlsson, T., and Jørgensen, J. L. (2012). Flight demonstration of non-cooperative rendezvous using optical navigation. In *23th International Symposium on Space Flight Dynamics*, pages 1–15.

Fehse, W. (2003). *Automated Rendezvous and Docking of Spacecraft*. Cambridge University Press, New York.

Felicetti, L., Gasbarri, P., Pisculli, A., Sabatini, M., and Palmerini, G. B. (2016). Design of robotic manipulators for orbit removal of spent launchers' stages. *Acta Astronautica*, 119:118–130.

Fischler, M. A. and Bolles, R. C. (1981). Random Sample Consensus: A paradigm for model fitting with applicatlons to image analysis and automated cartography. *Communications of the ACM*, 24(6):381 – 395.

Foix, S., Alenyà, G., and Torras, C. (2011). Lock-in Time-of-Flight (ToF) Cameras: A survey. *IEEE Sensors Journal*, 11(3):1–11.

Gomez, H. and Boge, T. (2015). Relative pose estimation of space debris using point cloud geometrical descriptors. In *3rd CEAS EuroGNC, Specialist Conference on Guidance, Navigation & Control*, pages 1–12.

Gómez Martínez, H., Giorgi, G., and Eissfeller, B. (2017). Pose estimation and tracking of non-cooperative rocket bodies using Time-of-Flight cameras. *Acta Astronautica*, 139:165–175.

Gomez Martinez, H. C. and Eissfeller, B. (2016). Autonomous determination of spin rate and rotation axis of rocket bodies based on point clouds. In *AIAA Guidance, Navigation, and Control Conference, AIAA SciTech Forum*, pages 1–13.

Gschwandtner, M., Kwitt, R., Uhl, A., and Pree, W. (2011). BlenSor: Blender sensor simulation toolbox. In *Advances in Visual Computing*, volume 6939, pages 199–208.

Hansard, M., Lee, S., Choi, O., and Horaud, R. (2013). *Time-of-Flight cameras - principles, methods and applications*. Springer-Verlag, London.

Horaud, R., Conio, B., Leboulleux, O., and Lacolle, B. (1989). An analytic solution for the perspective 4-point problem. In *Computer Vision and Pattern Recognition*, pages 500–507.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of ASME - Journal of Basic Engineering*, 82:35–45.

Kanani, K., Chabot, T., Petit, A., Marchand, E., and Gerber, B. (2012). Vision based navigation for debris removal missions. In *63rd International Astronautical Congress*, pages 1–8.

Kaufman, A., Cohen, D., and Yagel, R. (1993). Volume graphics. *Computer*, 26(7):51–64.

Kawamoto, S., Ohkawa, Y., Okamoto, H., Iki, K., Okumira, T., Katayama, Y., Hayashi, M., Horikawa, Y., Kato, H., Murakami, N., Yamamoto, T., and Inoue, K. (2017). Current status of research and development on active debris removal at JAXA. In *7th European Conference on Space Debris*, pages 1–7.

Keller, M. and Kolb, A. (2009). Real-time simulation of time-of-flight sensors. *Simulation Modelling Practice and Theory*, 17:967–978.

Kessler, D. J. and Cour-Palais, B. G. (1978). Collision frequency of artificial satellites: The creation of a debris belt. *Journal of Geophysical Research*, 83(A6):2637–2646.

Kim, J. and Rock, S. (2006). Stochastic feedback controller design considering the dual effect. In *AIAA Guidance, Navigation, and Control Conference*, pages 1–13.

Klasing, K., Althoff, D., Wollherr, D., and Buss, M. (2009). Comparison of surface normal estimation methods for range sensing applications. In *2009 IEEE International Conference on Robotics and Automation*, pages 3206–3211.

Klinkrad, H. (2006). *Space Debris - Models and Risk Analysis*. Springer-Verlag Berlin Heidelberg.

Kuipers, J. B. (1999). *Quaternions and Rotation Sequences*. Princeton University Press, Princeton.

Lappas, V. J., Forshaw, J. L., Visagie, L., Pisseloup, A., Salmon, T., Joffre, E., Chabot, T., Retat, I., Axthelm, R., Barraclough, S., Ratcliff, A., Bradford, A., Kadhem, H., Navarathinam, N., Rotteveel, J., Bernal, C., Chaumette, F., Pollini, A., and Steyn, W. H. (2014). RemoveDEBRIS: an EU low cost demonstration mission to test ADR technologies. In *65th International Astronautical Congress*, pages 1–12.

Liou, J.-C. (2017). Highlights of recent research activities at the NASA Orbital Debris Program Office. In *7th European Conference on Space Debris*, pages 1–7.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157.

Makarov, Y., Prokopchik, A., Simonov, M., Gorlov, A., Dubleva, A., Popkova, L., Stepanov, D., Loginov, S., Usovik, I., Yurash, V., and Yakovlev, M. (2017). Major trends for mitigation of space debris in near-Earth space in the Russian Federation. In *7th European Conference on Space Debris*, number April, pages 1–5.

MakerBot (2014). MakerBot Replicator Z18 3D Printer - User manual.

May, S., Droeschel, D., Holz, D., Wiesen, C., and Fuchs, S. (2008). 3D Pose estimation and mapping with Time-of-Flight cameras. In *IEEE/RSJ 2008 International Conference on Intelligent Robots and Systems*, pages 1–6.

NASA (2010). On-Orbit Satellite Servicing Study. Technical report, NASA.

National Research Council (1995). *Orbital Debris: A Technical Assessment*. The National Academies Press, Washington, DC.

Newman, P. and Ho, K. (2005). SLAM- Loop closing with visually salient features. In *IEEE International Conference on Robotics and Automation*, pages 635–642.

Ojakangas, G. W. and Cowardin, H. (2012). Probable rotation states of rocket bodies in Low Earth Orbit. In *13th annual Advanced Maui Optical and Space Conference*, pages 1–12.

Padial, J., Hammond, M., Augenstein, S., and Rock, S. M. (2012). Tumbling target reconstruction and pose estimation through fusion of monocular vision and sparse-pattern range data. In *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 419–425.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572.

Pele, O. and Werman, M. (2010). The quadratic-Chi histogram distance family. In *11th European Conference on Computer Vision*, volume 6312, pages 749–762.

Peters, S., Fiedler, H., and Förstner, R. (2015). ADReS-A: Mission architecture for the removal of SL-8 rocket bodies. In *IEEE Aerospace Conference*, pages 1–8.

PMD-Technologies (2010). PMD Vision CamCube 3.0 Datasheet.

Pollini, A., Blanc, N., and Mitev, V. (2011). Flash optical sensors for guidance, navigation and control systems. In *8th International Conference on Guidance, Navigation and Control Systems*, pages 1–7.

Ringbeck, T. and Hagebeuker, B. (2007). A 3D time of flight camera for object detection. In *Optical 3-D Measurements Techniques*, pages 1–10.

Rosso, F., Gallo, F., Allasia, W., Licata, E., Prinetto, P., Rolfo, D., Trotta, P., Favetto, A., Paleari, M., and Ariano, P. (2013). Stereo vision system for capture and removal of space debris. In *Design and Architectures for Signal and Image Processing*, pages 1–7.

Ruel, S., Luu, T., Anctil, M., and Gagnon, S. (2008). Target localization from 3D data for on-orbit autonomous rendezvous & docking. In *IEEE Aerospace Conference*, pages 1–11.

Rusu, R. B. (2010). *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Technische Universität München.

Rusu, R. B., Blodow, N., and Beetz, M. (2009). Fast Point Feature Histograms (FPFH) for 3D registration. In *IEEE International Conference on Robotics and Automation*, pages 3212–3217.

Rusu, R. B., Bradski, G., Thibaux, R., and Hsu, J. (2010). Fast 3D recognition and pose using the Viewpoint Feature Histogram. In *IIEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2155–2162.

Scharstein, D. and Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1):7–42.

Sellmaier, F., Boge, T., Spurmann, J., Gully, S., Rupp, T., and Huber, F. (2010). On-Orbit Servicing missions: Challenges and solutions for spacecraft operations. In *SpaceOps 2010 Conference*, pages 1–11.

Shan, M., Guo, J., and Gill, E. (2016). Review and comparison of active space debris capturing and removal methods. *Progress in Aerospace Sciences*, 80:18–32.

Soares Beleboni, M. G. (2014). A brief overview of Microsoft Kinect and its applications. In *Interactive Multimedia Conference 2014*, pages 1–6.

United Nations (1999). Technical Report on Space Debris. Technical report, United Nations, New York.

Woffinden, D. C. and Geller, D. K. (2007). Navigating the road to autonomous orbital rendezvous. *Journal of Spacecraft and Rockets*, 44(4):898–909.

Yanagisawa, T. and Kurosaki, H. (2012). Shape and motion estimate of LEO debris using light curves. *Advances in Space Research*, 50:136–145.

A | Quaternions

A quaternion is a 4-tuple of real numbers, discovered by William Rowan Hamilton in 1843 (Kuipers, 1999), and considered as a hyper-complex number. The latter consists of a scalar part and three orthogonal parts, commonly named as vectorial part. The principle of this complex number was based on the definition given by Hamilton:

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1 \tag{A.1}$$

where $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ are the orthonormal basis vectors for the euclidean space. From this rule, the quaternion can also be represented as:

$$\mathbf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} \tag{A.2}$$

The regular representation of a quaternion was already given in Equations 4.3 and 4.4, but included here again for reading easiness:

$$\mathbf{q} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{A.3}$$

$$\mathbf{q} = \begin{bmatrix} q_s \\ \mathbf{q}_v \end{bmatrix}; \qquad q_s = \begin{bmatrix} q_0 \end{bmatrix}, \qquad \mathbf{q}_v = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{A.4}$$

Due to the fact that the quaternion is a sum of a scalar with a vector, a new algebra had to be developed. For instance, the sum of two quaternions, $\mathbf{p}$ and $\mathbf{q}$, is defined

by adding the corresponding components, giving as a result a new quaternion, $\mathbf{s}$:

$$\mathbf{s} = \mathbf{p} + \mathbf{q} = (p_0 + q_0) + (p_1 + q_1)\mathbf{i} + (p_2 + q_2)\mathbf{j} + (p_3 + q_3)\mathbf{k} \tag{A.5}$$

The product of a quaternion and a scalar is the latter multiplied by each component of the former:

$$c\mathbf{q} = cq_0 + cq_1\mathbf{i} + cq_2\mathbf{j} + cq_3\mathbf{k} \tag{A.6}$$

Otherwise, the product of two quaternions heavily depends on the orthonormal vectors property given by Equation A.1, which leads to the following cases:

$$\begin{aligned}\mathbf{ij} &= -\mathbf{ji} = \mathbf{k}\\ \mathbf{ki} &= -\mathbf{ik} = \mathbf{j}\\ \mathbf{jk} &= -\mathbf{kj} = \mathbf{i}\end{aligned}$$

inducing to:

$$\begin{aligned}\mathbf{pq} &= (p_0 + p_1\mathbf{i} + p_2\mathbf{j} + p_3\mathbf{k})(q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k})\\ &= p_0q_0 - (p_1q_1 + p_2q_2 + p_3q_3)\\ &\quad + p_0(q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}) + q_0(p_1\mathbf{i} + p_2\mathbf{j} + p_3\mathbf{k})\\ &\quad + (p_2q_3 - p_3q_2)\mathbf{i} + (p_3q_1 - p_1q_3)\mathbf{j} + (p_1q_2 - p_2q_1)\mathbf{k}\end{aligned} \tag{A.7}$$

Using properties of the dot and cross products for vectors, and the quaternion's definition of $\mathbf{q} = q_s + \mathbf{q}_v$, i.e. Equation A.2, it is possible to write Equation A.7 in a more succinct form:

$$\mathbf{pq} = p_sq_s - \mathbf{p}_v \cdot \mathbf{q}_v + p_s\mathbf{q}_v + q_s\mathbf{p}_v + \mathbf{p}_v \times \mathbf{q}_v \tag{A.8}$$

The complex conjugate of the quaternion, $\mathbf{q}^*$, is defined as

$$\mathbf{q}^* = q_s - \mathbf{q}_v = q_0 - q_1\mathbf{i} - q_2\mathbf{j} - q_3\mathbf{k} \tag{A.9}$$

The squared norm of a quaternion is given by

$$\begin{aligned} \|\mathbf{q}\|^2 &= \mathbf{q}^*\mathbf{q} = \mathbf{q}\mathbf{q}^* \\ &= q_s^2 + \mathbf{q}_v \cdot \mathbf{q}_v \quad = q_0^2 + q_1^2 + q_2^2 + q_3^2 \end{aligned} \tag{A.10}$$

The inverse multiplicative of a quaternion is designated as

$$\mathbf{q}^{-1} = \frac{\mathbf{q}^*}{\|\mathbf{q}\|^2} \tag{A.11}$$

In case the quaternion is normalized, i.e. $\|\mathbf{q}\|^2 = 1$, then the inverse is equal to the conjugate, i.e. $\mathbf{q}^{-1} = \mathbf{q}^*$.

A.1 Quaternion Rotation Operator

The quaternion rotation operator, $L_q(\boldsymbol{v})$, acting on a vector $\boldsymbol{v}$, is defined as

$$L_q(\boldsymbol{v}) = \mathbf{q}^*\boldsymbol{v}\mathbf{q} \tag{A.12}$$

and represents a rotation of the vector $\boldsymbol{v}$ through an angle α about $\mathbf{q}_v$ as the axis of rotation, having as a condition that the quaternion $\mathbf{q}$ must be a *unit* quaternion, i.e. normalized, of the form

$$\mathbf{q} = q_s + \mathbf{q}_v = \cos\frac{\alpha}{2} + \mathbf{u}\sin\frac{\alpha}{2} \tag{A.13}$$

being $\mathbf{u}$ the unit vector which represents the direction of the quaternion.

A.2 Euler Angles to Quaternion

Taking again the definition of the Euler angles:

- ϕ is the roll angle
- θ is the pitch angle
- ψ is the yaw angle

the correspondent quaternion elements are:

$$\begin{aligned} q_0 &= \cos\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\phi}{2} + \sin\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\phi}{2} \\ q_1 &= \cos\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\phi}{2} - \sin\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\phi}{2} \\ q_2 &= \cos\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\phi}{2} + \sin\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\phi}{2} \\ q_3 &= \sin\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\phi}{2} - \cos\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\phi}{2} \end{aligned} \tag{A.14}$$

A.3 Quaternion to Euler Angles

The extraction of the Euler angles from a quaternion is given by the following trigonometric relationships:

$$\begin{aligned} \tan\phi &= \frac{m_{23}}{m_{33}} \\ \sin\theta &= -m_{13} \\ \tan\psi &= \frac{m_{12}}{m_{11}} \end{aligned} \tag{A.15}$$

where

$$\begin{aligned} m_{11} &= 2q_0^2 + 2q_1^2 - 1 \\ m_{12} &= 2q_1q_2 + 2q_0q_3 \\ m_{13} &= 2q_1q_3 - 2q_0q_2 \\ m_{23} &= 2q_2q_3 + 2q_0q_1 \\ m_{33} &= 2q_0^2 + 2q_3^2 - 1 \end{aligned} \tag{A.16}$$

In order to avoid ambiguities in the rotation sequence, the sign of the cosine of any Euler angle will be always positive by definition (Kuipers, 1999).