POLITECNICO MILANO 1863

School of industrial and information engineering

Department of Aerospace Science and Technologies - DAER

Master of Science in Space Engineering

VISUAL IMAGING FOR RELATIVE POSITION RECONSTRUCTION IN SATELLITES PROXIMITY OPERATIONS: PRELIMINARY ANALYSES

Giacomo Petrucci

916208

Supervisor: Prof. M. Lavagna

A.Y. 2020/2021

Abstract

During the course of time the continuous evolution of space missions required the development of more and more advanced technologies. In the recent years, autonomous spacecrafts, able to perform different kinds of mission, have been increasingly exploited. This type of spacecrafts are often equipped with optical sensors and a computer vision software , in such a way that even the most dangerous, but also the most interesting and useful, types of mission can be carried out, like docking, sampling of an asteroids, relative navigation, etc...
Computer vision is the field dealing with optical navigation, and it involves, as the name implies, the understanding of how a computer see and elaborate an image. It is based on many technologies, statistics, signal and image processing, optimization methods etc... .
A common and popular process by which visual navigation is carried out is the V-SLAM, Visual-Simultaneous Location And Mapping, which, as stated by the name, is able to estimate the location of a robot, equipped with a camera, in an unknown environment and simultaneously mapping the latter.
In this thesis the V-SLAM algorithm, in one of its many forms, has been implemented and investigated, in order to see if a real time implementation could be possible, and its accuracy. The architecture selected is that of a monocular camera applied to the space sector. Due to the versatility of the instrument, both for the performance and the economic side, a synthetic image generator software, capable of reproducing almost any scenario, has been exploited to create images of an unknown, for now, and uncooperative object. Due to the vast number of scenario it could not be possible to perform tests on everyone of them, so one has been chosen, proximity operations/active space debris removal. In the last year the worldwide interest for this new type of mission has been exponentially increasing and, for this reason, it has been selected. The target chosen is a space debris called VESPA. Then, after the creation of the images, a feature based method for relative navigation has been implemented and tests have been carried out. In conclusion, the results obtained are presented and critically analysed, giving also some tracks to follow in order to improve their quality.

Keywords: Relative navigation, Optical navigation, Active space debris removal, Computer vision, SLAM, ORB-SLAM, synthetic images, Blender.

Sommario

Durante il corso del tempo la continua evoluzione delle missioni spaziale ha richiesto lo sviluppo di tecnologie sempre più avanzate. Negli ultimi anni, spacecrafts autonomi capaci di eseguire differenti tipi di missioni, sono stati utilizzati sempre di più. Questa tipologia di spacecraft sono spesso equipaggiati con dei sensori ottici ed un software di computer vision, così che anche le tipologie di missioni più pericolose, ma allo stesso tempo più interessanti ed utili, possano essere svolte, per esempio, attracco, raccolta di campioni da un asteroide, navigazione relativa, ecc...
Il campo della computer vision è quello che si occupa della navigazione ottica, e, come si può evincere dal nome, cerca di capire come un computer vede ed elabora le immagini. Si basa su molte tecnologie, statistica, elaborazione di immagini e di segnali, metodi d'ottimizzazione, ecc... . Un processo comune e popolare con il quale è spesso svolta la navigazione ottica, è lo V-SLAM, Visual-Simultaneous Location And Mapping, che, come possibile capire dal nome, si occupa di stimare la posizione di un robot, dotato di una camera, in un ambiente sconosciuto e simultaneamente mappare quest'ultimo.
In questa tesi l'algoritmo dello V-SLAM, in una delle sue tante forme, è stato implementato ed investigato, così da vedere se un suo utilizzo in tempo reale sia possibile, nonchè anche la sua accuratezza. L'archietettura selezionata è quella di una camera monoculare nel settore spaziale. Grazie alla versatilità dello strumento, sia dal lato delle prestazione che da quello economico, un software in grado di generare immagini sintetiche, fedeli quasi a tutti i tipi di scenario, è stato utilizzato per creare immagini di un oggetto sconosciuto, per il momento, e non cooperativo. Poiché il numero di scenari possibili è molto vasto, non è possibile fare dei test su ogni di essi, uno di loro è stato scelto, operazione di prossimità/Rimozione attiva di detriti spaziali. L'interesse del mondo in questo argomento è aumentato esponenzialmente negli ultimi anni e per questo è stato selezionato. L'oggetto scelto come obbiettivo è un detrito spaziale di nome VESPA. Quindi, dopo la creazione delle immagini, un metodo per la navigazione relativa basato sulle features è stato implemetato e dei tests sono stati svolti. In conclusione, i risultati ottenuti sono presentati e analizzati con occhio critico, dando anche dei consigli da seguire in futuro per migliorarli.

Parole chiavi: Navigazione relativa, Navigazione ottica, Rimozione attiva di detriti spaziali, Computer vision, V-SLAM, ORB-SLAM, Immagini sintetiche, Blender.

Ringraziamenti

Prima di proseguire con la trattazione dell'elaborato, mi sembra doveroso spendere qualche riga per ringraziare tutte le persone che hanno reso tutto questo, chi in un modo chi nell'altro, possibile.

In primo luogo vorrei ringraziare la Professoressa Michèle Lavagna per avermi dato l'opportunità di lavorare su questa tesi, che con le sue sfide e le sue difficoltà mi ha permesso di acquisire conoscenze riguardo nuovi argomenti, andando così ad allargare e ad arricchire la mia formazione qui al Politecnico di Milano.

Inoltre vorrei ringraziere tutti i miei amici, sia vecchi che nuovi, per avermi fatto compagnia e a volte anche "costretto" a studiare durante questi anni. In futuro ricorderò con malinconia l'estati passate in taverna a studiare con Michele e Simone, dove una pausa di 10 minuti si trasformava in un torneo di 1 ora a ping pong, le ore passate in biblioteca a studiare con Francesco, Gloria, Stella, Marco, Gianluca e Paolo, i giorni passati al politecnico a seguire le lezioni dove, fra una chiacchierata ed un'equazione, il tempo volava. Una menzione speciale va fatta per Gianluca e Paolo, poiché con loro ho condiviso la maggior parte di questo percorso, mi hanno sempre spinto a dare il massimo ed erano sempre lì pronti ad aiutarmi. Gianluca è tutt'ora il miglior coinquilino che abbia mai avuto, condividevamo e condividiamo tutt'ora le stesse passioni, dentro e fuori il poli, ed è grazie a lui se ho iniziato ad appassionarmi alla palestra, del resto, come dicevano i latini, "mens sana in corpore sano". A Paolo, o meglio, ai suoi appunti, devo tutto, senza di loro preparare gli esami sarebbe stato molto più difficile e lungo. Infine voglio ringraziare anche tutti i "ragazzi di moscova" per le bellissime serate passate in questi anni.

In ultimo, ma assolutamente non per importanza, vorrei ringraziare tutta la mia famiglia per essermi stata sempre accanto, la zia Patrizia, che fino all'ultimo si è preoccupata di come stesse andando questo mio percorso, lo zio Francesco ed i suoi gelati post-pranzo, le mie cugine Mariacristina e Francesca, loro mi hanno insegnato che bisogna sempre andare avanti con il sorriso, anche se la vita fa di tutto per togliertelo. Un grazie ai miei fratelli

Paolo e Stefano, che sono stati e sono tutt'ora i miei modelli di riferimento. La nonna Adele, la mitica ed inarrestabile nonna Adele, sempre pronta a cucinarmi qualsiasi cosa le chiedessi perchè alla fine, la cosa importante, era se io avessi mangiato oppure no, i polpettoni che mi facevi riportare qui a Milano rimarranno per sempre nel mio cuore. In fine un grazie enorme ai miei genitori, Rocco e Manola, sempre presenti e comprensivi, mi hanno sempre permesso di seguire le mie passioni senza mai farmi pesare nulla. Non me ne voglia Papà ma la mamma è sempre la mamma, magari a volte un po' oppressiva e pressante, ma del resto, quale mamma non lo è? Si è preoccupata sempre di tutto, e di più, non lasciando mai nulla al caso, qualsiasi cosa chiedessi arrivava subito, aveva una soluzione ad ogni problema ed è stata sempre la mia sostenitrice numero uno.

Grazie a tutti, davvero.

Contents

List of Figures

List of Tables

Acronyms

DOF	Degree Of Freedom
S/C	Spacecraft
ECI	Earth Centered Inertial
LVLH	Local Vertical Local Horizontal
SCI	Sun Centered Inertial
CW	Clohessy-Wiltshare
CM	Center of Mass
FOV	Field of VieW
ORB	Oriented FAST and Rotated Brief
ISS	International Space Station
VO	Visual Odometry
SLAM	Simultaneous Localization and Mapping
KF	Kalman Filter
EKF	Extended Kalman Filter
ROI	Region of Interest
VESPA	VEga Secondary Payload Adapter
CFRP	Carbon Fiber Reinforced Polymer
AU	Astronomical Unit
fps	frame per second
BU	Blender Unit
FS	frame-step
RMSE	Root Mean Square Error
SVD	Single Value Decomposition
DLT	Direct Linear Transform
GNC	Guidance.Navigation and Control

1 Introduction

Since the dawn of times, any being starts taking its first steps into the world by mapping it and then it moves accordingly, in such a way it doesn't hit any object/obstacle and hurt itself. This process can be defined as *relative navigation*. In space, this concept can be expressed as the problem of estimating the relative position and orientation of one object, typically a S/C, called deputy with respect to another one called chief, the latter can be any space object, comet, S/C, asteroids, etc. Like said before, this is a fundamental process for any entity, starting from the simplest life forms and finishing with rovers exploring new planets, in particular their surface, so of course, satellites are not an exception, and the necessity to have some sort of localization procedures is more important now than ever. Obviously, these kind of procedures already exist, but they are mostly remote ones, nowadays the concept of autonomy, having on-board awareness, is emerging and a lot of interest is put into it, especially for missions involving formation flying and proximity operations since time critical decision must be made in such a situation.

Two object orbiting together can be categorized into two classes:

- **Cooperative**: The two object can communicate to each other, a telecommunication link is established and the two object share the appropriate pose information. This class is mainly composed of S/C since instruments needed to reconstruct the state are necessary and they must be active, in fact each S/C is tasked to reconstruct its state and then share it with the other/others.
- **Uncooperative and unrelated**: As the name implies, the two object don't cooperate, like for example a S/C and a comet. In this class, typically, the S/C is used to study/acquire information about the other object.

In the case of the second class, the question of how to reconstruct the distance and the pose of the object of interest is fundamental, laser or camera based sensor are a common choice to answer that question. A *LiDAR*, Light Detection and Ranging, is, as can be deduced from the name, a radar that exploits light, a laser, in order to obtain the range of the target object. Through the spinning and the tilting of the laser beam, a full scan of the environment can be performed too. Although these solutions have an high degree of accuracy, and so can be applied to the problem of relative navigation with respect to another satellite or space object [1], they are very expensive. On the other hand, cameras offer simple and low cost solutions, but with a reduced accuracy of the results. If the latter solution is chosen, the saved resources could be redirected towards achieving other mission objectives. Camera can be used in two configurations, *mono* and *stereo*, the former is very appealing for cubesats, while the latter is used in order to obtain the depth of the scene. The two sensor described above can be also used together in order to indirectly obtain the scene depth.

As said before, the concept of autonomy is exponentially gaining interest, this is due to the fact that it will grant great improvements to the current generation of space mission and it will also affect the future one. As said in [2], autonomy is the ability of a system to achieve goals operating independently of external control, so it require *self-directedness*, to achieve its goal, and *self-sufficiency*, to operate independently.

In the Space sector this can affect many different fields, spanning from spacecraft formation flying to rover ground operation, Fig. 1.1.

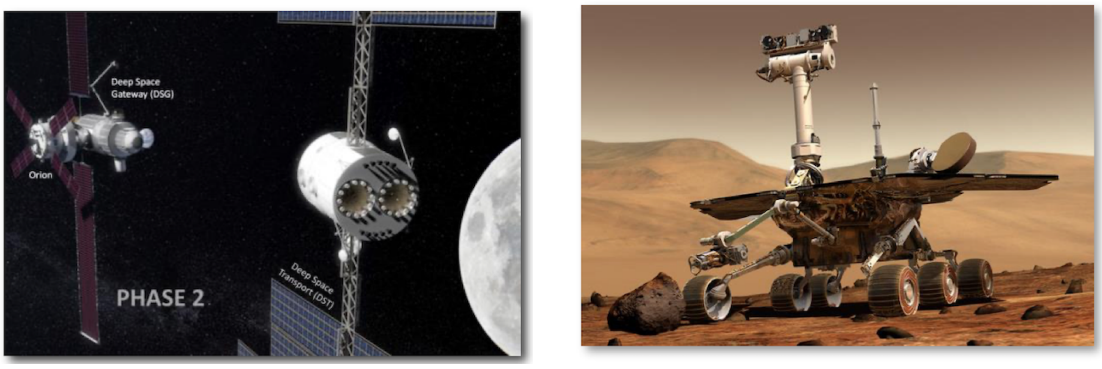


Figure 1.1: *Autonomous relative navigation applicable fields*

In order to perform close relative navigation and subsequently, if requested, proximity operations in a safe way, precise pose knowledge is necessary. This new approach can be applied to a lot of actions, starting from the ones with cooperative class objects, like, docking, refueling, satellite inspection and maintenance, to the ones with unknown and uncooperative objects, like, space debris removal, asteroid navigation and mining. Although the introduction of autonomy could boost the performance of a space system, it must not be taken lightly since an high risk is involved in its implementation.

1.1 State of the Art: Relative navigation in space

During the course of time, from the first space mission on wards,a lot of them were developed with different forms of relative navigation, here, some of them will be briefly discussed. A lot of technology demonstration mission have seen the light due to the increased interest in autonomous proximity operations.

Talking first about missions with cooperative objects, and starting from one of the most recent and prominent one, docking with the ISS, International Space Station, has become an important problem to solve and in 2009 a solution based on LiDAR has been proposed by NASA and the Canadian Space Agency. An all-in-one solution, called TriDAR (Triangulation & Light Detection and Ranging Automated Rendezvous & Docking), it is an automated rendezvous and docking sensor composed by a LiDAR and a thermal imager. The shape of the target is assumed to be known and is compared to the estimated one to retrieve important parameters like the relative range and position between the spacecraft and the ISS, up to 1 km distance. Three Space Shuttle missions and Cygnus spacecraft adopted this technology [3]. Also Dragon spacecraft uses a similar sensor asset called DragonEye. Relative GPS navigation is used up to 750 m, after which the ISS is tracked using the LiDAR, DragonEye, and thermal cameras [4].

Going back in time, in 2005 NASA launched two S/Cs, XSS-11 and DART [5]. The first was meant to conduct proximity operations with its carrier, the second stage of the Minotaur I rocket, while the second should have performed a 2d hours mission with no human intervention, in which several rendezvous were included, but unfortunately it ended with a soft collusion with its target due to a malfunction of the spacecraft. Going further back, in 1997, the first mission for demonstration of rendezvous and docking capabilities was launched, it was the Japanese ETS-VII [6]. Proximity operations were carried out automatically and with remote control through a camera-aided robotic arm and in orbit refuelling was also investigated successfully.

Regarding relative navigation around unknown and uncooperative space object, the main and most recent mission are Hayabusa 2 [7], OSIRIS-Rex [8] and Rosetta [9]. Since the set-up of the mission is similar to the one under the study of this thesis, few camera parameters are listed too.

Rosetta was lunched in 2004, it consisted of an orbiter and a lander for the first ever study of a cometa, the 67P/Churyumov-Gerasimenko. It was equipped a redundant navigation camera called NAVCAM [10] for AOCS only, and a pair of camera for optical, infrared and spectroscopic imaging called OSIRIS [11]. The imaging data necessary for the optical navigation of the S/C around the comet were provided by the NavCam. It had a CCD sensor resolution of 1024x1024 px and a focal length of 152.5 mm with an aperure of 30 or 70 mm, depending on the mode. Regarding OSIRIS, it had two cameras, a high resolution Narrow angle camera, NAC, with a CCD detectors of 2048x2048 px, that, as stated before, was used to produce high resolution images, with a focal length of 717 mm and an aperture of 90 mm, giving a f_{number} of, almost, 8, and a Wide Angle Camera, WAC, mainly used to study the intensity of gas emissions and dust-scattered sunlight, it had a focal length of around 140 mm.

Talking about the other two missions cited before, they are both an asteroid sample and return, the first, Hayabusa-2, from JAXA (Japanese Space Exploration Agency) developed to study Asteroid 1999 JU3 [12], while the second, OSIRIS-REx, was developed by NASA to study asteroid Bennu [13], and, in case of necessity, asteroid 1999 JU3 as backup. The latter is provided with a redundant LiDAR for ranging to the surface and two NavCams [14], used to support the navigation during various mission operations, in particular during the sampling operation. The images obtained from them are used to track the star-fields and landmarks on Bennu in order to determine the spacecraft position. Going back to Hayabusa-2, it had a LiDAR and three Optical Navigation Cameras, two them are wide angle cameras and are called ONC-W1 and ONC-W2, the last one is a telescopic camera and is called ONC-T. All of them use a CCD detectors of size 1024x1024 px and the telescopic one has a focal leght of 121 mm and an aperture of 15 mm.

As can be seen from the cases presented, these missions generally use a mix of laser-based and vision-based sensor for relative navigation. In particular, for Rosetta, visual navigation was fundamental for the whole mission duration. When in the far approach navigation, images were used to understand the centroid position of the comet and perform optical measurements. In the following phase of comet characterization and mapping a visual navigation based on landmarks was performed on ground, a manual process at first and automatic later on, it allowed to predict the S/C position and orientation along with a very detailed shape of the comet.

These system however had limitations, such as the amount of data that could be sent due to the link budget, thus limiting the possibilities for proximity operations. This problems could be solved by researching and developing more autonomous navigation system.

1.2 Computer vision: Visual relative navigation methods

Computer vision is very vast and multidisciplinary fields, it is tasked with the mission of understanding and extract useful information from images. Though at the beginning as an easy problem to solve, it proved to be a very complex challenge and it still is. The field in which development is nowadays more innovative is that of robotics, it contains the best tolls for estimating the trajectory of a camera. Due to the numerous way this technology can be used, it is becoming increasingly popular even in the space context, in fact, a wide range of missions like, in-orbit rendezvous and docking, asteroid and small body sampling operations and planetary approach and landing, use it.

During the years numerous algorithms were developed in the various sectors, but fundamentally they are based on two methods: Visual Odometry, VO, and Simultaneous Location and Mapping, SLAM. Starting from the latter, as stated in "Simultaneous localization and mapping: Part I" [15]:

"SLAM is a process by which a mobile robot can build a map of an environment and at the same time use this map to deduce its location."

Since the argument of the discussion is computer vision, the SLAM problem is usually referred as V-SLAM problem where the V stands for Visual. There are mainly two way to approach and solve the online, real time version of it. The first is based on the use of filtering techniques, the most popular being the ones of the Kalman filter family (KF,EKF,...). They form a toll that continuously predicts and update the state of a dynamical system in time recursively. Due to the fact that they are very light in terms of memory and are fast, they are the ideal tool for real time problems and embedded system for space applications [16] [17].

The second one is based on the keyframe method, it is popular due to the optimization approach of global bundle adjustment. It is an optimization algorithm for structure and motion refinement but ut us usually too expensive, in terms of memory and velocity, to be executed in real time. To overcome this hurdle, it is possible to use this algorithm on a subset of the available frames, also called keyframes. They are selected from the available frames and are mainly used to reduce the computational load and enable real time performances. As for the first way, also here there are popular algorithm, one of them is the ORB-SLAM [18], it is a complex workflow, as can be seen in Fig. 1.2 capable of automatic map initialization, loop detection and closure, re-localization.
To see which of the way gives better result, a sturdy to campare the two has been performed, in [19], and it concluded that keyframe-based techniques with bundle adjustment are better than the filtering ones for the same computational cost, but, if small processing budget is available, the last one might be better.

For space application, solutions include the use of ORB-SLAM for S/C non-cooperative rendezvous image sequences, such as in [20]. It performs in an accurate way in very different scenarios, differently from the robotic fields. It must be said, as written in the previously cited paper, that pose reconstruction is lost when too many features disappear from the images, but the algorithm is always able to recover the trajectory once it revisits previously known map features.
According to "ORB-SLAM: a Versatile and Accurate Monocular SLAM System" [21]:

"(...) no other system has demonstrated to work as in many different scenarios and with such accuracy. Therefore our system is currently the most reliable and complete solution for monocular SLAM."

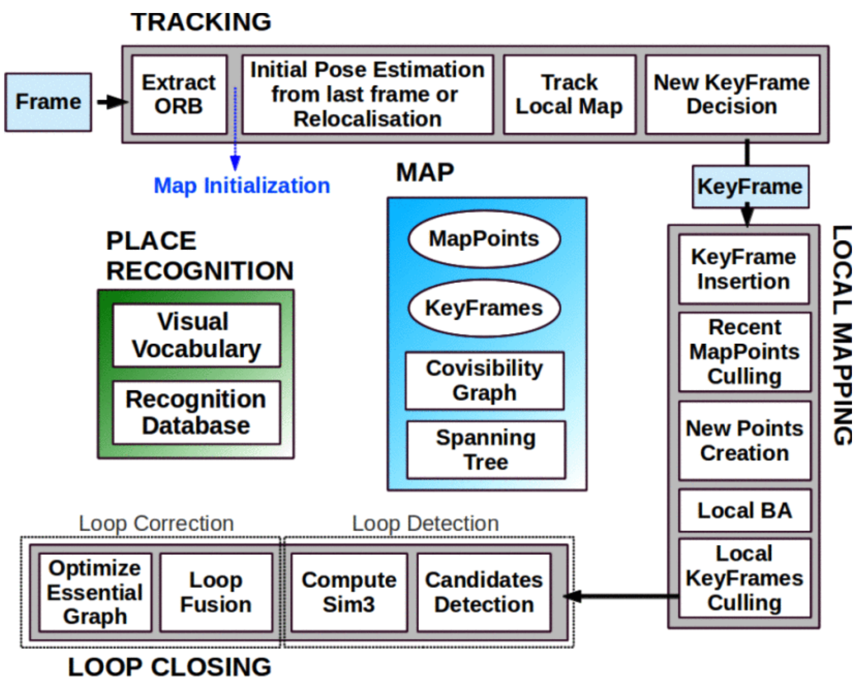


Figure 1.2: *ORB-SLAM workflow [21]*

Going to the Visual Odometry, as reported in "Visual Odometry [Tutorial]" [22]:

"VO is the process of estimating the egomotion of an agent (e.g. vehicle, human, robot) using only the input of a single or multiple cameras."

The operational concept is to examine how the motion of the vehicle generates changes in images, thus reconstructing the pose of the camera, so of the vehicle itself, in an incremental way. It can be considered a subcategory of the V-SLAM problem. The main difference between the two methods is that the latter, VO, focuses on local consistency while the former focuses on the global consistency of the trajectory. The feature that differentiate them is the one on *loop closure*, performed in the V-SLAM. It consists on the recognition of features from area already visited, in order to correct the entire list of stored features and thus improve the accuracy of the results. Due to these characteristics different methodologies are use to solve the VO problem and it might be useful mainly in situation where it would be impossible, or improbable, to revisit a particular 3D scene point twice, such as landing trajectory or rover ground operations, in fact, the Mars Exploration mission [23], exploits the VO to support attitude and position estimation. Since it's possible that not every part of the image is in the interest of the task that must be performed, and therefore should not be processed in a V-SLAM framework, a feature that might be needed in a vision-based algorithm is the one of ROI, Region of Interest, estimation. It can be the case of a spacecraft image with the Earth in the background, in fact, the different motion of the target S/C and the background can be very dangerous for the mapping and trajectory estimation process.

1.3 Rendering for space application

As for every new (and old) technology, everything must be first tested, for that generations of synthetic images became increasingly important, as a tool to support physical tests.

There are two main research areas:

- Simulation of the surface of celestial bodies.
- Simulation of images from sensors, like navigation cameras.

The testing of these systems, even in terrestrial environment, is very expensive and complicated [24]. Four main problems have been encountered in the use of physical models [25]:

- *Illumination*, using lamps to recreate the different kind of lighting that can be present during a mission and, the movement of light during its various phases, can be quite challenging.
- *Model Calibration and Measurement*, the calibration of the test instrumentation can be quite difficult and, not only the miscellaneous tools, but also the various physical models require attention in this regard.
- *Low flexibility in changing the scenario*, as the name implies, this is related to the difficulty of changing the scenario to find out how robust an algorithm is. Since modifying the first model would be difficult, it will be necessary to create a new one, process that turns out to be time consuming and, more important, expensive.
- *Scalability and Resolution*, as before, the problem can be inferred from the name, due to the available space, it is almost impossible to built the model in 1:1 scale. When recreating a model in scale it is required that the accuracy of the details will not be altered so, manufacturing constraints are present, and the model can't be scaled at will.

Even if important problems are present, the use of laboratory tests is still fundamental since this is the only way in which, for example, test real sensor with real noise. The use of a rendering software can help all these procedures, but not substitute them, indeed, it can be an useful instrument to deploy in combination with a GNC facility.

Imaging software for the creation of synthetic image, even in a space environment, already exist and are available online. One of the is **SurRender** [26], Fig. 1.3, developed by Airbus Defence & Space. It utilizes a ray tracing engine in order to generate images specifically for space scenarios, such as planetary approach, landing and in orbit rendezvous. The software is free to use upon request.

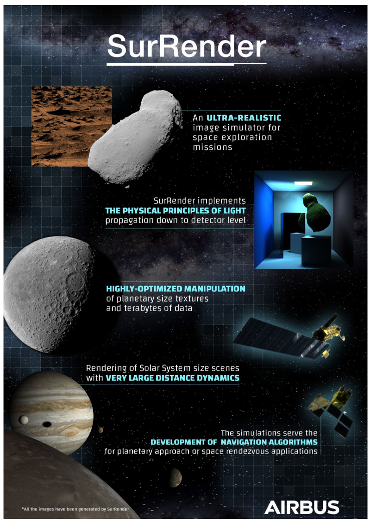

SurRender
An ULTRA-REALISTIC image simulator for space exploration missions
SurRender implements THE PHYSICAL PRINCIPLES OF LIGHT propagation down to detector level
HIGHLY-OPTIMIZED MANIPULATION of planetary size textures and terabytes of data
Rendering of Solar System size scenes with VERY LARGE DISTANCE DYNAMICS
The simulations serve the DEVELOPMENT OF NAVIGATION ALGORITHMS for planetary approach or space rendezvous applications
*All the images have been generated by SurRender
AIRBUS

Figure 1.3: *SurRender Software*

Another similar software is **PANGU** [27], Fig. 1.4, Planet and Asteroid Natural Scene Generation Utility, developed by the University of Dundee in Scotland with ESA's support. It is able to generate camera and LiDAR images of different planetary bodies and spacecraft to test vision-guided navigation, guidance and landing system. The software is licensed and to be used free of charges in ESA projects only.

Figure 1.4: *Planet and Asteroid Natural Scene Generation Utility*

Both of them include various effects, some induced by the space environment and some that can happen everywhere, like, lens distortion, internal light scattering, motion blur, electronic noises, rolling shutter etc.

The software used in this thesis to obtain synthetic images of a spacecraft is **Blender** [28], Fig. 1.5. It is a free and open source 3D creation suite, very popular in the entertainment industry, often used in animation, film production and 3D modeling.

Figure 1.5: *Blender software*

1.4 Thesis overview

The aim of this thesis is to explore the topic of vision-based real time relative navigation between two unknown and uncooperative space objects in order to check if it is possible to apply it to future missions, such as active space debris removal, planetary landing, sampling, etc. In order to do so, one of the previous scenario has been selected, active space debris removal, and a target has been chosen, VESPA. Blender is used to develop realistic space images of it, implementing data from a real camera. The calibration of its parameters is then performed, as should be done for any real mission. A V-SLAM like method has been implemented in Matlab and in order to replicate as close as possible the actual scenario of a mission, where the computational power of a CPU is, usually, not very high, the CPU of the used computer, a 2,2 GHz Intel Core i7 quad-core, has been capped at 40% of its full potential.

1.4.1 Thesis structure

Regarding the structure of the thesis, it is as follow:

- Chapter 2

  It exposes some basic concepts of theory concerning relative dynamics, then going into more specific cases.mainly investigating the system of equations for nonlinear translational-rotational relative dynamics.

- Chapter 3

  In this section projective geometry theory and the basic camera model are discussed.

It starts with the basis of homogeneous transformations and finsh with the image formations through a pinhole camera model and the calibration of a camera.

- Chapter 4
  It contains an insight into the field of computer vision (theory and method), starting from feature extraction, feature matching and tracking and then epipolar geometry. It also includes important numerical methods like fundemental matrix estimation, triangulation and PnP, Perspective-n-Point.

- Chapter 5
  In this section the workflow of the thesis and its set-up steps are presented. Starting from the target identification, the creation of its model in blender, the process to follow in order to calibrate the parameters of the camera and ending with the presentation of the V-SLAM algorithm implemented.

- Chapter 6
  It shows the results of the analysis: first the parameters obtained from the camera calibration are reported and after, the analysis of the V-SLAM results are discussed.

- Chapter 7
  The conclusion of the work and its future developments are here reported.

2 Relative Dynamics

The main task of an autonomous navigation system is, as the name implies, to make navigation around the specified target possible; for this reason, Relative Dynamics has been one of the topics first investigated. The goal of any computer vision algorithm is the one of reconstructing it from images, estimating how the target appears in the different frames.

It is imperative to remember that the environment in which the mission is taking place is space and the relative dynamics in this kind of scenario is very different, for example, from those present in the field of robotics. During the passage of time several models have been developed, each one with its own limitations and constraints, depending on the characteristic of the orbit of the target, the perturbations considered and even the available computing power, therefore, it is not easy to establish the most suitable one.

The models present in literature range over the simplest and most common ones, based on point masses [29], to the more complex and accurate ones able to describe the relative motion of a 6 DOF spacecraft.

In the following sections the nonlinear relative dynamics system of equations and the coupled translational-rotational [30] relative dynamics system of equations are derived, but, before jumping in these more complex topics, a brief description of the general orbital dynamics model, and the different kind of reference frames used in space, are presented.

2.1 Orbital Dynamics and Reference frames

The two-body problem equations of motion in an inertial reference frame are given by:

$$\ddot{\mathbf{r}} = -\mu \frac{\mathbf{r}}{r^3} \tag{2.1}$$

where $\mathbf{r}$ is the position of the S/C and μ is the standard gravitational parameter, described as μ= GM, in which, G is Newton's constant of gravitation and M is the mass of the body the S/C is orbiting. It's important to point out that in this description of the equation of motion a simplification has been made, the mass of the S/C has been neglected, since, usually, it's various order of magnitude less than the one of the main body, in this case planets.

Another way to describe the orbit of the S/C is by using the keplerian elements:

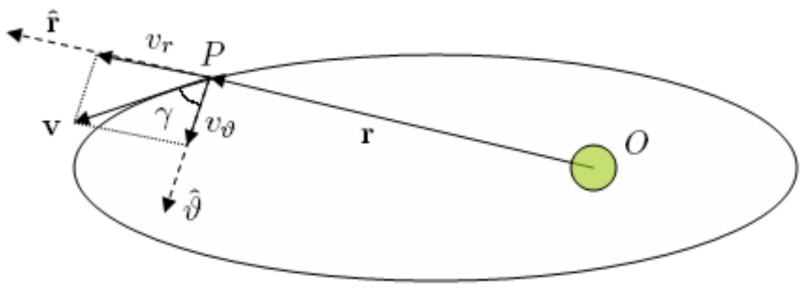


Figure 2.1: *Position and velocity along the orbit*

- $\boldsymbol{\alpha}$ = semi-major axis
- $\mathbf{e}$ = eccentricity
- $\mathbf{i}$ = inclination
- $\boldsymbol{\Omega}$ = Right Ascension of the Ascending node
- $\boldsymbol{\omega}$ = argument of perigee
- $\boldsymbol{\nu}$ = True anomaly

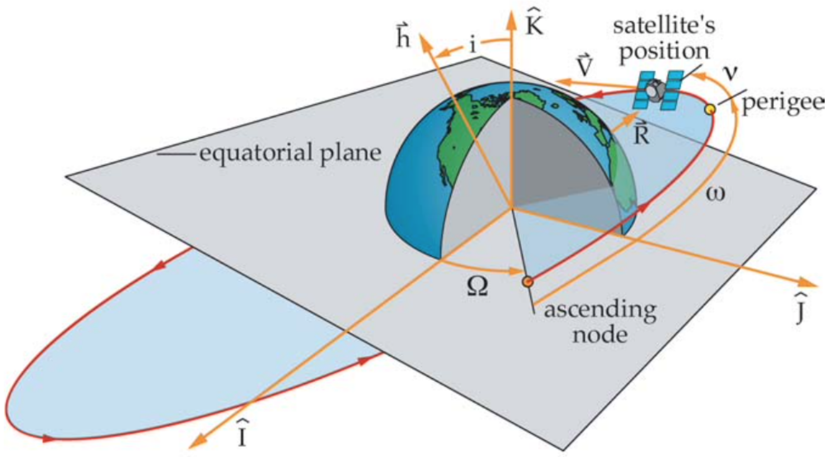


Figure 2.2: *Graphic representation of four of the six keplerian elements [31]*

A graphic representation of the last four keplerian elements can be seen in fig. 2.2. Regarding $\boldsymbol{\alpha}$ and $\mathbf{e}$, they are the classic parameter of any ellipses.
Since this is a general and basic topic for the sector, it won't be shown how to derive these elements from the position and velocity of the S/C.

2.1.1 Reference frames

In this section some standard reference frames are described since most of them are used through this chapter.

- **ECI**, Earth Centered Inertial reference frames [I](Fig. 2.3): Inertial reference frame where **X** points towards the vernal equinox and forms the equatorial plane with **Y**. **Z** is positive towards the North Pole.

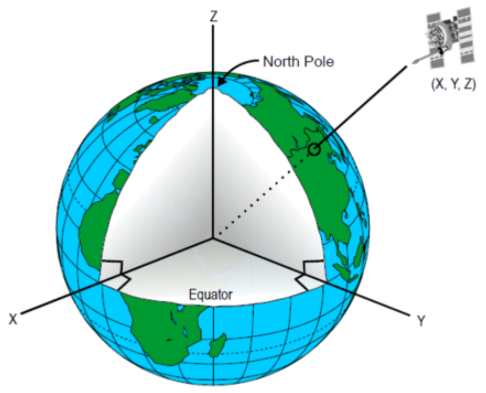


Figure 2.3: *Earth Centered Inertial reference frames [32]*

- **SCI** Sun Centered Inertial reference frames [S](Fig. 2.4): Inertial reference frame where $\mathbf{X_s}$ points towards the vernal equinox, $\mathbf{Z_s}$ is normal to the ecliptic plane and points towards the Northern Hemisphere and $\mathbf{Y_s}$ is perpendicular to the $\mathbf{X_s}$ and $\mathbf{Z_s}$ axes, forming a right-handed coordinate system.

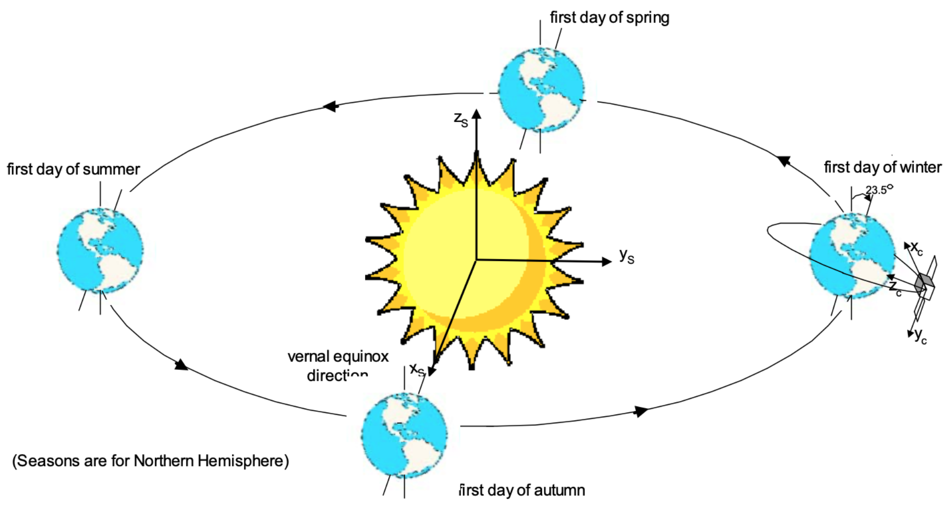


Figure 2.4: *Sun Centered Inertial reference frame [33]*

- Perifocal reference frame [P](Fig. 2.5): $\overline{\mathbf{x}}$ and $\overline{\mathbf{y}}$ denote the orbital plane, with $\overline{\mathbf{x}}$ pointing towards the periapsis of the orbit, and $\overline{\mathbf{z}}$ perpendicular to the orbital plane, forming a right-handed coordinate system.

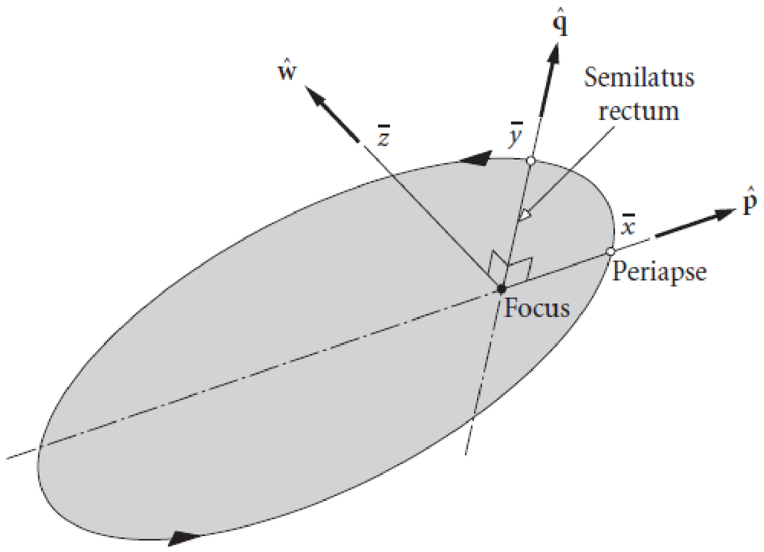


Figure 2.5: *Perifocal reference frame [34]*

- LVLH, Local Vertical Local Horizontal reference frame [L](Fig. 2.6): $\mathbf{x}$ is directed as the radial vector, $\mathbf{y}$ lies in the orbital plane and $\mathbf{z}$ completes the frame.

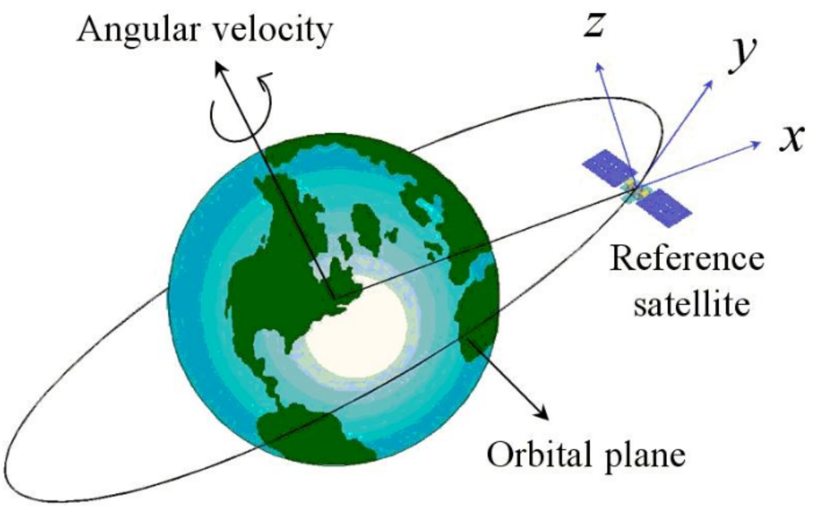


Figure 2.6: *LVLH reference frame for the relative motion [35]*

- Rotating reference frame [R](Fig. 2.7) used to write equations in polar coordinates. The unit vector $\hat{\mathbf{r}}$ is directed from the primary radially outward, and the angle θ is measured in the counterclockwise direction from some reference line, PQ, the line of nodes in the Earth case, to $\mathbf{r}$. The angles ω and f are also shown.

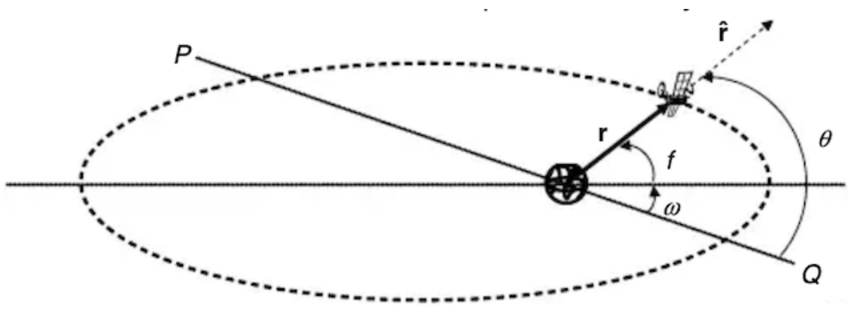


Figure 2.7: *Rotating reference frame [36]*

2.2 System of equation for nonlinear relative dynamics

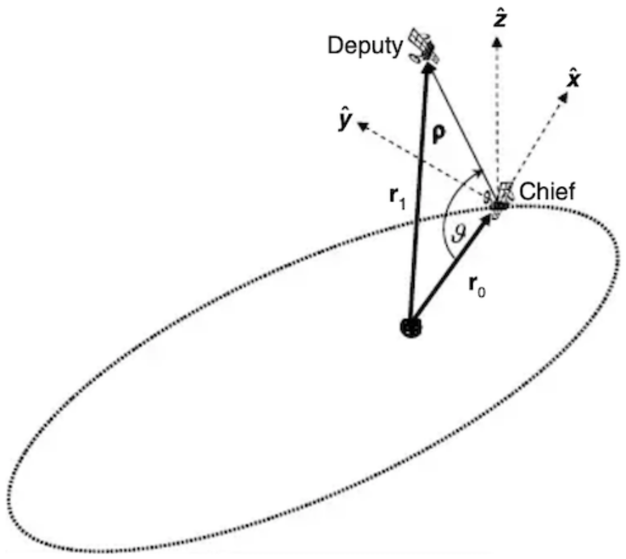


Figure 2.8: *Chief and Deputy S/C [36]*

As described in eq. 2.1, the two-body problem equations of motion in an inertial reference frame are:

$$\ddot{\mathbf{r}} = -\mu \frac{\mathbf{r}}{r^3} \tag{2.2}$$

and by expressing them for the chief and the deputy they become:

$$\ddot{\mathbf{r_c}} = -\mu \frac{\mathbf{r_c}}{{r_c}^3} \tag{2.3}$$

$$\ddot{\mathbf{r}}_{\mathbf{d}} = -\mu \frac{\mathbf{r}_{\mathbf{d}}}{{r_{\mathrm{d}}}^3} \tag{2.4}$$

where, referring to Fig. 2.8, $\mathbf{r_c} = \mathbf{r_0}$ and $\mathbf{r_d} = \mathbf{r_1}$.

The relative position $\boldsymbol{\rho}$ of the deputy with respect to the chief one, as can be seen from fig. 2.8, is:

$$\boldsymbol{\rho} = \mathbf{r_d} - \mathbf{r_c} \tag{2.5}$$

Starting from Eq. 2.3 and subtracting Eq. 2.4, then defining the position of the deputy with respect to the chief as $\mathbf{r_d}$= $\mathbf{r_c}$ + $\boldsymbol{\rho}$, the following expression is obtained:

$$\ddot{\boldsymbol{\rho}} = -\frac{\mu}{||\mathbf{r_c} + \boldsymbol{\rho}||}(\mathbf{r_c} + \boldsymbol{\rho}) + \mu \frac{\mathbf{r_c}}{{r_{\mathrm{c}}}^3} \tag{2.6}$$

This discussion started from the ECI reference frame so, an inertial one, however it would be useful to have the equation written with respect to a non inertial reference frame, like the LVLH one. The relative acceleration $\ddot{\boldsymbol{\rho}}$ in the LVLH frame can be recovered starting from [37]:

$$\mathbf{a_a} = \mathbf{a_r} + \mathbf{a}_\tau + \mathbf{a_c} \tag{2.7}$$

where $\mathbf{a_a}$ is the absolute acceleration of the deputy measured in [I], $\mathbf{a_r}$ is $\ddot{\boldsymbol{\rho}}$, so the relative acceleration of the deputy with respect to the chief, measured in [L], $\mathbf{a}_\tau$ = $\mathbf{a_{Ch}}$ + $\dot{\boldsymbol{\omega}} \times (\mathbf{r_c}$ - $\mathbf{r_d})$ + $\boldsymbol{\omega} \times (\boldsymbol{\omega} \times (\mathbf{r_c}$ - $\mathbf{r_d}))$ is the entrained acceleration and $\mathbf{a_c}$ = $2\boldsymbol{\omega} \times \mathbf{v_r}$ is the Coriolis acceleration. $\boldsymbol{\omega}$ is the relative angular velocity of the LVLH frame with respect to the ECI one, so the difference between their angular velocity; $\mathbf{v_r}$ is the relative velocity of the deputy (relative to the chief) $\dot{\boldsymbol{\rho}}$, so meausered in [L]. Substituing all the terms yelds:

$$\mathbf{a_a} - \mathbf{a_{Ch}} = \mathbf{a_r} + \dot{\boldsymbol{\omega}} \times (\mathbf{r_c} - \mathbf{r_d}) + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times (\mathbf{r_c} - \mathbf{r_d})) + 2\boldsymbol{\omega} \times \mathbf{v_r} \tag{2.8}$$

since $\mathbf{a_a}$ - $\mathbf{a_{Ch}}$ is the relative acceleration of the deputy with respect to the chief measured in [I], and making the derivative explicit, Eq. 2.8 can be rewritten as:

$$\frac{d^{2\mathrm{I}}\boldsymbol{\rho}}{d^2t} = \frac{d^{2\mathrm{L}}\boldsymbol{\rho}}{d^2t} + \frac{d\,{}^{\mathrm{I}}\boldsymbol{\omega}^{\mathrm{L}}}{dt} \times \boldsymbol{\rho} + {}^{\mathrm{I}}\boldsymbol{\omega}^{\mathrm{L}} \times ({}^{\mathrm{I}}\boldsymbol{\omega}^{\mathrm{L}} \times \boldsymbol{\rho}) + 2\,{}^{\mathrm{I}}\boldsymbol{\omega}^{\mathrm{L}} \times \frac{d^{\mathrm{L}}\boldsymbol{\rho}}{dt} \tag{2.9}$$

The notation ${}^{\mathrm{I}}\boldsymbol{\omega}^{\mathrm{L}}$ denotes the difference between the angular velocity of the two frames [L] and [I], $\boldsymbol{\omega}_{\mathrm{L}}$ - $\boldsymbol{\omega}_{\mathrm{I}}$. Since [I] is an inertial reference frames, its angular velocity $\boldsymbol{\omega}_{\mathrm{I}}$ is null, so the derivative of ${}^{\mathrm{I}}\boldsymbol{\omega}^{\mathrm{L}}$ measured from the inertial frame [I] or the non inertial one [L], is the same, for this reason, its derivative doesn't have any apexes, differently from the ones of $\boldsymbol{\rho}$.

Substituting Eq. 2.6 and the quantities of $\boldsymbol{\omega}$, $\mathbf{r_c}$ and $\boldsymbol{\rho}$ in [L], ${}^{\mathrm{I}}\boldsymbol{\omega}^{\mathrm{L}}$ = $[0,0,\dot{\theta}]^{\mathrm{T}}$, $\mathbf{r_c}$=

$[r_c,0,0]^T$, $\boldsymbol{\rho} = [x,y,z]^T$ into Eq. 2.9, the system of nonlinear equations for relative motion is finally derived [36]:

$$\begin{cases} \ddot{x} - 2\dot{f}_c\dot{y} - \ddot{f}_c y - \dot{f}_c^2 x = -\frac{\mu(r_c+x)}{[(r_c+x)^2+y^2+z^2]^{\frac{3}{2}}} + \frac{\mu}{r_c^2} \\ \ddot{y} - 2\dot{f}_c\dot{x} - \ddot{f}_c x - \dot{f}_c^2 y = -\frac{\mu y}{[(r_c+x)^2+y^2+z^2]^{\frac{3}{2}}} \\ \ddot{z} = -\frac{\mu z}{[(r_c+x)^2+y^2+z^2]^{\frac{3}{2}}} \end{cases} \tag{2.10}$$

This set of equation alone is not sufficient to describe the full motion of the deputy, the equation for absolute motion of the chief, Eq. 2.3, is needed. In Eq. 2.10 the terms f_c (chief true anomaly) and r_c appear, they can be obtained solving Eq. 2.3 in [R], where $\mathbf{r_c}$= r_c $\hat{\mathbf{r}}_c$ derivating two $\hat{\mathbf{r}}_c$=$\hat{\mathbf{r}}_c$ (t), it is then obtained:

$$\begin{cases} \ddot{r}_c = r_c\dot{f}_c^2 - \frac{\mu}{r_c^2} \\ \ddot{f}_c = -\frac{2\dot{r}_c\dot{f}_c}{r_c} \end{cases} \tag{2.11}$$

Eq. 2.10 and Eq. 2.11 form the complete set needed to obtain the full motion of the deputy, they form a system of five second order differential equation which can be reduced in order into the ODEs and then integrated.

2.2.1 Perturbations and Control actions

Up until now, no perturbation **d** and no control forces **u** have been considered, to take them into account it is necessary to add some terms to Eq. 2.10, resulting in a not so different system of equations:

$$\begin{cases} \ddot{x} - 2\dot{f}_c\dot{y} - \ddot{f}_c y - \dot{f}_c^2 x = -\frac{\mu(r_c+x)}{[(r_c+x)^2+y^2+z^2]^{\frac{3}{2}}} + \frac{\mu}{r_c^2} + d_x + u_x \\ \ddot{y} - 2\dot{f}_c\dot{x} - \ddot{f}_c x - \dot{f}_c^2 y = -\frac{\mu y}{[(r_c+x)^2+y^2+z^2]^{\frac{3}{2}}} + d_y + u_y \\ \ddot{z} = -\frac{\mu z}{[(r_c+x)^2+y^2+z^2]^{\frac{3}{2}}} + d_z + u_z \end{cases} \tag{2.12}$$

As can be seen in the system above, the only differences, with respect to Eq. 2.10, are the components of the perturbations and control forces.

2.2.2 Linearized model, the Clohessy-Wiltshare Equations

Even if the motion of the deputy is described by a set of non linear equations, under determined conditions, the system can be linearized and so simplified, this is exactly the case for the Clohessy-Wiltshare equations [36].

The conditions necessary to apply this method are:

– The chief's orbit must be circular.

– The deputy's orbit in [I] is only slightly elliptical and slightly inclined with respect to the chief's one.

The more these conditions are respected, the more the two models (nonlinear e linear) overlap, and, provided that the initial conditions are first order small, the motion of the deputy will appear very close to the chief in a chief-fixed frame.

Since the chief's orbit is circular it is possible to write $r_c = \alpha_c$, where α_c is the semi-major axis, or, in this case, radius, of the orbit. Substituting it to Eq. 2.10 and expanding the right-hand side into a Taylor series about the origin and taking only the first-order terms:

$$\begin{cases} -\frac{\mu(\alpha_c+x)}{[(\alpha_c+x)^2+y^2+z^2]^{\frac{3}{2}}} \approx n_0^2(2x-\alpha_c) \\ -\frac{\mu y}{[(\alpha_c+x)^2+y^2+z^2]^{\frac{3}{2}}} \approx -n_0^2 y \\ -\frac{\mu z}{[(\alpha_c+x)^2+y^2+z^2]^{\frac{3}{2}}} \approx -n_0^2 z \end{cases} \tag{2.13}$$

where $n_0 = \sqrt{\frac{\mu}{\alpha_c^3}}$.

Rearranging the terms, the set of equations (CW) that governs the relative motion of the deputy with respect to chief is obtained:

$$\begin{cases} \ddot{x} - 2n_0\dot{y} - 3n_0^2 x = 0 \\ \ddot{y} + 2n_0\dot{x} = 0 \\ \ddot{z} + n_0^2 z = 0 \end{cases} \tag{2.14}$$

Solving this set of equations is straightforward and easier than solving Eq. 2.10.

It must be noted that the method discussed above is just one of many that can be used to obtain the CW equations.

2.3 System of equations for coupled translational-rotational relative dynamics

The case described above consider the two S/Cs, the chief and the deputy, as two point masses, but, when the two are in close proximity and their relative distance is similar in order of magnitude to their dimensions, they cannot be modelled as such. It is necessary to take into account both the relative dynamic and the attitude of the chief and the deputy, and develop a model accordingly.

Each S/C will have its own LVLH frame, [L$_\text{c}$] and [L$_\text{d}$], its own Body frame, [B$_\text{c}$] and

$[\mathrm{B_d}]$. For the model discussed in the following sections it is assumed that the satellites are Earth pointing, so that $[\mathrm{L_c}] \equiv [\mathrm{B_c}]$ and $[\mathrm{L_d}] \equiv [\mathrm{B_d}]$.

2.3.1 Relative rotational model

Defining the rotational angular velocity of the deputy relative to the chief as:

$$\boldsymbol{\omega} = \boldsymbol{\omega}_d - \boldsymbol{\omega}_c \tag{2.15}$$

.

where $\boldsymbol{\omega}_\mathrm{c}$ and $\boldsymbol{\omega}_\mathrm{d}$ are, respectively, the angular velocity of the chief and the deputy in some given reference frame.

It's possible to parameterize the relative attitude using a rotation matrix $\mathbf{D}$ (3×3) that transform a vector from the body-fixed frame $[\mathrm{B_d}]$ to the body-fixed frame $[\mathrm{B_c}]$. This parametrization will be performed by using the Euler parameters $\mathbf{e}$, eigenaxis of rotation, and $\boldsymbol{\gamma}$, the eigenangle of rotation, which uniquely parametrize the Euler rotation from a reference frame to another.The Euler paramenters form a quaternion $\mathbf{q} = [q_1, q_2, q_3, q_4]^T$:

$$\begin{cases} q_1 = e_1 sin(\frac{\gamma}{2}) \\ q_2 = e_2 sin(\frac{\gamma}{2}) \\ q_3 = e_3 sin(\frac{\gamma}{2}) \\ q_4 = cos(\frac{\gamma}{2}) \end{cases} \tag{2.16}$$

where e_1, e_2, e_3 are the component of the eigenaxis of rotation $\mathbf{e}$.

The rotation matrix $\mathbf{D}$ expressed in terms of quaternions becomes:

$$\mathbf{D}(\mathbf{q}) = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 - q_3q_4) & 2(q_1q_3 + q_2q_4) \\ 2(q_1q_2 + q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 - q_1q_4) \\ 2(q_1q_3 - q_2q_4) & 2(q_2q_3 + q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \tag{2.17}$$

Using the matrix above, the relative angular velocity vector $\boldsymbol{\omega}$ can be calculated in $[\mathrm{B_c}]$, remembering that $[\mathrm{L_c}] \equiv [\mathrm{B_c}]$, as follow:

$$\boldsymbol{\omega}|_{\mathrm{L_c}} = \boldsymbol{\omega}_d|_{\mathrm{L_c}} - \boldsymbol{\omega}_c|_{\mathrm{L_c}} = \mathbf{D}(\mathbf{q})\boldsymbol{\omega}_d|_{\mathrm{L_d}} - \boldsymbol{\omega}_c|_{\mathrm{L_c}} \tag{2.18}$$

Utilizing $\boldsymbol{\omega}$ and $\mathbf{q}$, the attitude kinematic of the deputy relative the chief can be described using the quaternion kinematic equation of motion:

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{Q}(\mathbf{q})\boldsymbol{\omega}|_{\mathrm{L_d}} \tag{2.19}$$

with:

$$\mathbf{Q}(\mathbf{q}) = \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_4 & -q_3 & q_2 \\ q_3 & q_4 & -q_1 \\ -q_2 & q_1 & q_4 \end{bmatrix} \tag{2.20}$$

To derive the attitude dynamics of the deputy relative to the chief it is first necessary to differentiate Eq. 2.15 in the inertial frame [I]:

$$\frac{d^{\mathrm{I}}\boldsymbol{\omega}}{dt} = \frac{d^{\mathrm{I}}\boldsymbol{\omega}_d}{dt} - \frac{d^{\mathrm{I}}\boldsymbol{\omega}_c}{dt} \tag{2.21}$$

Now, expressing it in [L$_{\mathrm{c}}$] yields:

$$\frac{d^{\mathrm{I}}\boldsymbol{\omega}}{dt}|_{\mathrm{L_c}} = \mathbf{D}(\mathbf{q})\frac{d^{\mathrm{I}}\boldsymbol{\omega}_d}{dt}|_{\mathrm{L_d}} - \frac{d^{\mathrm{I}}\boldsymbol{\omega}_c}{dt}|_{\mathrm{L_c}} \tag{2.22}$$

On the other hand, it is possible to obtain the derivative of $\boldsymbol{\omega}|_{\mathrm{L_c}}$ in [I] following the classic derivation rule from an inertial fram [I] to a non-inertial frame [V] of a vector [37]:

$$\frac{d^{\mathrm{I}}\boldsymbol{\omega}}{dt} = \frac{d^{\mathrm{L_c}}\boldsymbol{\omega}}{dt} + \boldsymbol{\omega}_c \times \boldsymbol{\omega} \tag{2.23}$$

Expressing it in [L$_{\mathrm{c}}$] provides:

$$\frac{d^{\mathrm{I}}\boldsymbol{\omega}}{dt}|_{\mathrm{L_c}} = \frac{d^{\mathrm{L_c}}\boldsymbol{\omega}}{dt}|_{\mathrm{L_c}} + \boldsymbol{\omega}_c|_{\mathrm{L_c}} \times \boldsymbol{\omega}|_{\mathrm{L_c}} \tag{2.24}$$

Comparing Eq. 2.22 and Eq. 2.24 yields:

$$\frac{d^{\mathrm{L_c}}\boldsymbol{\omega}}{dt}|_{\mathrm{L_c}} = \mathbf{D}(\mathbf{q})\frac{d^{\mathrm{I}}\boldsymbol{\omega}_d}{dt}|_{\mathrm{L_d}} - \frac{d^{\mathrm{I}}\boldsymbol{\omega}_c}{dt}|_{\mathrm{L_c}} - \boldsymbol{\omega}_c|_{\mathrm{L_c}} \times \boldsymbol{\omega}|_{\mathrm{L_c}} \tag{2.25}$$

Eq. 2.25 relates the derivative of the relative angular velocity to the angular velocity rates of the deputy and the chief, thus yielding the relative rotational dynamic equations, but, the objective of this discussion is to have them expressed only by the relative angular velocity and the angular velocity rate of the chief, thus dropping the one of the deputy. In order to do that several passages must be followed.

The first one uses the fact that the differentiation of the angular velocity in a fixed frame or in a body frame gives the same result, so Eq. 2.25 becomes:

$$\frac{d^{\mathrm{L_c}}\boldsymbol{\omega}}{dt}|_{\mathrm{L_c}} = \mathbf{D}(\mathbf{q})\frac{d^{\mathrm{L_d}}\boldsymbol{\omega}_d}{dt}|_{\mathrm{L_d}} - \frac{d^{\mathrm{L_c}}\boldsymbol{\omega}_c}{dt}|_{\mathrm{L_c}} - \boldsymbol{\omega}_c|_{\mathrm{L_c}} \times \boldsymbol{\omega}|_{\mathrm{L_c}} \tag{2.26}$$

Then it is necessary to multiply Eq. 2.26 by the Inertia Tensor of the chief, $\mathbf{I}_{\mathrm{c}}$.

Now, starting from the total angular momentum $\mathbf{H}$, it is known that:

$$\frac{d\mathbf{H}}{dt} = \mathbf{N} \tag{2.27}$$

where $\mathbf{N}$ is the total external torque, if any.
Applying the Eq. 2.23, both for the chief and the deputy, and writing $\mathbf{H}_\mathrm{c}$ as $\mathbf{I}_\mathrm{c}\boldsymbol{\omega}_\mathrm{c}$ and $\mathbf{H}_\mathrm{d}$ as $\mathbf{I}_\mathrm{d}\boldsymbol{\omega}_\mathrm{d}$, with $\mathbf{I}_\mathbf{d}$ being the Inertia Tensor of the deputy, yields:

$$\begin{cases} \mathbf{I}_d \dfrac{d^{\mathrm{I}}\boldsymbol{\omega}_d}{dt} = \mathbf{I}_d \dfrac{d^{\mathrm{L_d}}\boldsymbol{\omega}_d}{dt} + \boldsymbol{\omega}_d \times \mathbf{I}_d\boldsymbol{\omega}_d = \mathbf{N}_d \\ \mathbf{I}_c \dfrac{d^{\mathrm{I}}\boldsymbol{\omega}_c}{dt} = \mathbf{I}_c \dfrac{d^{\mathrm{L_c}}\boldsymbol{\omega}_c}{dt} + \boldsymbol{\omega}_c \times \mathbf{I}_c\boldsymbol{\omega}_c = \mathbf{N}_c \end{cases} \tag{2.28}$$

Now, substituting Eq. 2.28 into Eq. 2.26, after multiplying it by $\mathbf{I}_\mathrm{c}$, and then using Eq. 2.18, yields the equation for the relative attitude dynamics expressed using relative and $[\mathrm{L_c}]$-related angular velocities:

$$\begin{aligned} \mathbf{I}_c \frac{d^{\mathrm{L_c}}\boldsymbol{\omega}}{dt}\Big|_{\mathrm{L_c}} =& \mathbf{I}_c\mathbf{D}(\mathbf{q})\mathbf{I}_d^{-1}\{\mathbf{N}_d - \mathbf{D}(\mathbf{q})^T(\boldsymbol{\omega}|_{\mathrm{L_c}} + \boldsymbol{\omega}_c|_{\mathrm{L_c}}) \times \mathbf{I}_d\mathbf{D}(\mathbf{q})^T(\boldsymbol{\omega}|_{\mathrm{L_c}} + \boldsymbol{\omega}_c|_{\mathrm{L_c}})\} \\ & - \mathbf{I}_c\boldsymbol{\omega}_c|_{\mathrm{L_c}} \times \boldsymbol{\omega}|_{\mathrm{L_c}} - \{\mathbf{N}_c - \boldsymbol{\omega}_c|_{\mathrm{L_c}} \times \mathbf{I}_c\boldsymbol{\omega}_c|_{\mathrm{L_c}}\} \end{aligned} \tag{2.29}$$

Eq. 2.19 and Eq. 2.29 fully describe the relative attitude behaviour.

2.3.2 Relative translational model

The coupling between the rotational and translation dynamics arises because of either external torques (the most obvious example is the gravity gradient torque, which depends on altitude) or internal coupling, which is external-perturbations independent. The internal coupling comes from the fact that relative motion equations can be written for any point on the S/C, not necessarily the CM, since as said before, they can't be considered anymore as point masses. Thus, an apparent translational motion of points on the deputy S/C (that do not coincide with the CM) will result from rotation of the deputy about its CM, so feature points of the two S/C must be taken into consideration.

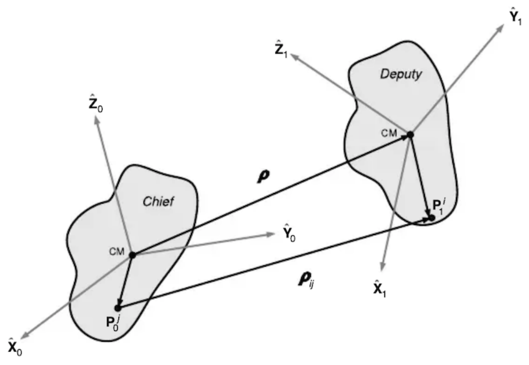


Figure 2.9: *Chief and deputy S/C, two rigid-body with body-fixed reference frame [36]*

Looking at Fig. 2.9 it must be clarified that the notation used refer $\mathbf{P}_0$ to $\mathbf{P}_\mathrm{c}$ and $\mathbf{P}_1$ to $\mathbf{P}_\mathrm{d}$, and the same for all the other components.

It is important to remark that the S/C in this discussion are assumed to be rigid. Now, defining $\mathbf{P}_{\mathrm{c}}{}^{\mathrm{j}}$ as a vector directed from the origin of the coordinate system [$\mathrm{L_c}$], so the CM of the chief, to the point P_c^j, and $\mathbf{P}_{\mathrm{d}}{}^{\mathrm{i}}$ as a vector directed from the origin of the coordinate system [$\mathrm{L_d}$], so the CM of the deputy, to the point P_d^i, as shown in Fig. 2.9, the relative position of point j of the chief and point i of the deputy can be defined as $\boldsymbol{\rho}_{\mathrm{ij}}$. $\boldsymbol{\rho}$ is the same as before.
By observing Fig. 2.9 it is also possible to note that the following relationship holds:

$$\mathbf{P}_c^j + \boldsymbol{\rho}_{\mathrm{ij}} = \boldsymbol{\rho} + \mathbf{P}_d^i \tag{2.30}$$

thus:

$$\boldsymbol{\rho}_{\mathrm{ij}} = \boldsymbol{\rho} + \mathbf{P}_d^i - \mathbf{P}_c^j \tag{2.31}$$

Taking the first and second-order derivative it's obtained:

$$\dot{\boldsymbol{\rho}}_{\mathrm{ij}} = \dot{\boldsymbol{\rho}} + \dot{\mathbf{P}}_d^i - \dot{\mathbf{P}}_c^j \tag{2.32}$$

$$\ddot{\boldsymbol{\rho}}_{\mathrm{ij}} = \ddot{\boldsymbol{\rho}} + \ddot{\mathbf{P}}_d^i - \ddot{\mathbf{P}}_c^j \tag{2.33}$$

where in frame [$\mathrm{L_c}$]:

$$\frac{d^{\mathrm{L_c}}\mathbf{P}_c^j}{dt}|_{\mathrm{L_c}} = \frac{d^{2\mathrm{L_c}}\mathbf{P}_c^j}{dt^2}|_{\mathrm{L_c}} = 0 \tag{2.34}$$

while, calculating $\dot{\mathbf{P}}_d^i$ and $\ddot{\mathbf{P}}_d^i$ with respect to the rotating frame [$\mathrm{L_c}$] yields:

$$\frac{d^{\mathrm{L_c}}\mathbf{P}_d^i}{dt}|_{\mathrm{L_c}} = \frac{d^{\mathrm{L_d}}\mathbf{P}_d^i}{dt}|_{\mathrm{L_d}} + \boldsymbol{\omega} \times \mathbf{P}_d^i \tag{2.35}$$

$$\frac{d^{2\mathrm{L_c}}\mathbf{P}_d^i}{dt^2}|_{\mathrm{L_c}} = \frac{d^{2\mathrm{L_d}}\mathbf{P}_d^i}{dt^2}|_{\mathrm{L_d}} + 2\boldsymbol{\omega} \times \frac{d^{\mathrm{L_d}}\mathbf{P}_d^i}{dt}|_{\mathrm{L_d}} + \dot{\boldsymbol{\omega}} \times \mathbf{P}_d^i + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{P}_d^i) \tag{2.36}$$

and, as happened before for the chief, since the deputy is considered as a rigid body:

$$\frac{d^{\mathrm{L_d}}\mathbf{P}_d^i}{dt}|_{\mathrm{L_d}} = \frac{d^{2\mathrm{L_d}}\mathbf{P}_d^i}{dt^2}|_{\mathrm{L_d}} = 0 \tag{2.37}$$

This leads us to the following equations:

$$\frac{d^{\mathrm{L_c}}\boldsymbol{\rho}_{\mathrm{ij}}}{dt}|_{\mathrm{L_c}} = \frac{d^{\mathrm{L_c}}\boldsymbol{\rho}}{dt}|_{\mathrm{L_c}} + \boldsymbol{\omega} \times \mathbf{P}_d^i \tag{2.38}$$

$$\frac{d^{2\mathrm{L_c}}\boldsymbol{\rho}_{\mathrm{ij}}}{dt^2} = \frac{d^{\mathrm{L_c}}\boldsymbol{\rho}}{dt}|_{\mathrm{L_c}} + \dot{\boldsymbol{\omega}} \times \mathbf{P}_d^i + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{P}_d^i) \tag{2.39}$$

Then, expressing the vectors using their components, $\boldsymbol{\rho} = [x, y, z]^T$, $\boldsymbol{\rho}_{\mathrm{ij}} = [x_{\mathrm{ij}}, y_{\mathrm{ij}}, z_{\mathrm{ij}}]^T$,$\mathbf{P}_d^i = [P_{\mathrm{x_d}}{}^i, P_{\mathrm{y_d}}{}^i, P_{\mathrm{z_d}}{}^i]^T$, $\mathbf{P}_c^j = [P_{\mathrm{x_c}}{}^j, P_{\mathrm{y_c}}{}^j, P_{\mathrm{z_c}}{}^j]^T$,$\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^T$ and substituting them first into Eqs. 2.31,2.38,2.39 and then substituting again into Eq. 2.10, the model describing the translational motion between any arbitrary point $\mathbf{P}_{\mathrm{c}}{}^{\mathrm{j}}$ and $\mathbf{P}_{\mathrm{d}}{}^{\mathrm{i}}$, in the absence of any perturbation or control forces, is obtained:

$$\begin{cases}
\ddot{x}_{\mathrm{ij}} - [\omega_y(\omega_x P_{\mathrm{y_d}}{}^i - \omega_y P_{\mathrm{x_d}}{}^i) + \omega_z(\omega_z P_{\mathrm{x_d}}{}^i - \omega_x P_{\mathrm{z_d}}{}^i)] - \dot{\omega}_y P_{\mathrm{z_d}}{}^i + \dot{\omega}_z P_{\mathrm{y_d}}{}^i \\
-2\dot{f}_c[y_{\mathrm{ij}} - (\omega_z P_{\mathrm{x_d}}{}^i - \omega_x P_{\mathrm{z_d}}{}^i)] - \ddot{f}_c(y_{\mathrm{ij}} - P_{\mathrm{y_d}}{}^i + P_{\mathrm{y_c}}{}^j) - \dot{f}_c^2(x_{\mathrm{ij}} - P_{\mathrm{x_d}}{}^i + P_{\mathrm{x_c}}{}^j) \\
= \frac{-\mu(r_c + x_{\mathrm{ij}} - P_{\mathrm{x_d}}{}^i + P_{\mathrm{x_c}}{}^j)}{[(r_c + x_{\mathrm{ij}} - P_{\mathrm{x_d}}{}^i + P_{\mathrm{x_c}}{}^j)^2 + (y_{\mathrm{ij}} - P_{\mathrm{y_d}}{}^i + P_{\mathrm{y_c}}{}^j)^2 + (z_{\mathrm{ij}} - P_{\mathrm{z_d}}{}^i + P_{\mathrm{z_c}}{}^j)^2]^{\frac{3}{2}}} + \frac{\mu}{r_c^2} \\
\ddot{y}_{\mathrm{ij}} - [\omega_z(\omega_y P_{\mathrm{z_d}}{}^i - \omega_z P_{\mathrm{y_d}}{}^i) + \omega_x(\omega_x P_{\mathrm{y_d}}{}^i - \omega_y P_{\mathrm{x_d}}{}^i)] - \dot{\omega}_z P_{\mathrm{x_d}}{}^i + \dot{\omega}_x P_{\mathrm{z_d}}{}^i \\
+2\dot{f}_c[x_{\mathrm{ij}} - (\omega_y P_{\mathrm{z_d}}{}^i + \omega_z P_{\mathrm{y_d}}{}^i)] + \ddot{f}_c(x_{\mathrm{ij}} - P_{\mathrm{x_d}}{}^i - P_{\mathrm{x_c}}{}^j) - \dot{f}_c^2(y_{\mathrm{ij}} - P_{\mathrm{y_d}}{}^i + P_{\mathrm{y_c}}{}^j) \\
= \frac{-\mu(y_{\mathrm{ij}} - P_{\mathrm{y_d}}{}^i + P_{\mathrm{y_c}}{}^j)}{[(r_c + x_{\mathrm{ij}} - P_{\mathrm{x_d}}{}^i + P_{\mathrm{x_c}}{}^j)^2 + (y_{\mathrm{ij}} - P_{\mathrm{y_d}}{}^i + P_{\mathrm{y_c}}{}^j)^2 + (z_{\mathrm{ij}} - P_{\mathrm{z_d}}{}^i + P_{\mathrm{z_c}}{}^j)^2]^{\frac{3}{2}}} \\
\ddot{z}_{\mathrm{ij}} - [\omega_x(\omega_z P_{\mathrm{x_d}}{}^i - \omega_x P_{\mathrm{z_d}}{}^i) + \omega_y(\omega_y P_{\mathrm{z_d}}{}^i - \omega_z P_{\mathrm{y_d}}{}^i)] - \dot{\omega}_x P_{\mathrm{y_d}}{}^i + \dot{\omega}_y P_{\mathrm{x_d}}{}^i \\
= \frac{-\mu(z_{\mathrm{ij}} - P_{\mathrm{z_d}}{}^i + P_{\mathrm{z_c}}{}^j)}{[(r_c + x_{\mathrm{ij}} - P_{\mathrm{x_d}}{}^i + P_{\mathrm{x_c}}{}^j)^2 + (y_{\mathrm{ij}} - P_{\mathrm{y_d}}{}^i + P_{\mathrm{y_c}}{}^j)^2 + (z_{\mathrm{ij}} - P_{\mathrm{z_d}}{}^i + P_{\mathrm{z_c}}{}^j)^2]^{\frac{3}{2}}}
\end{cases} \tag{2.40}$$

These equations are coupled to the rotational motion equations, Eqs. 2.19, 2.29, through the components of the relative angular velocity vector $\boldsymbol{\omega}$, and, together, they form a set of 15 ODEs, since f_c is already inside Eq. 2.40 to be integrated. These are only for the relative motion of the deputy with respect to the chief, so in order to have the full system, it is necessary to include Eq. 2.11, that describe the absolute motion of the chief, in particular only the ODEs for $\mathbf{r}_c$ are needed, bringing the total of the ODEs to be integrated to 17. Even in this case a CW linearization can be applied.

Even though these models have not been used later on, they offer a valuable framework in which develop this thesis. Furthermore, most rendendering software adopt the same quantities definition, and since they also allow the definition of a trajectory of an object before imaging, such models can be used to automate the trajectory rendering process. Moreover, relative motion models are fundamental when a filter based solution to the V-SLAM problem is sought.

3 Camera

One of the most important concepts of an autonomous navigation system is the one of, as the name implies, autonomy. In order to achieve that, the S/C must be able to perceive its surroundings, this can be done through the use of sensors. There are multiple kind of sensors but, in a mission where optical navigation is required, cameras are essential.

In the following chapter the camera model and the context in which cameras operate and how pictures are taken will be discussed.

Starting from the former, Euclidean geometry cannot fully explain it. For example, two parallel lines in $\mathbb{R}^3$ will never intersect, however, as can be seen from our daily life, things are perceived differently. Looking at Fig. 3.1, it is rapidly noticed that the two rails, that in reality would never intersect, meet each other in a point at infinity. This phenomenon, in art, is called perspective [38], and a way to describe it, so how 3D objects are projected in 2D images, comes from projective geometry [39].

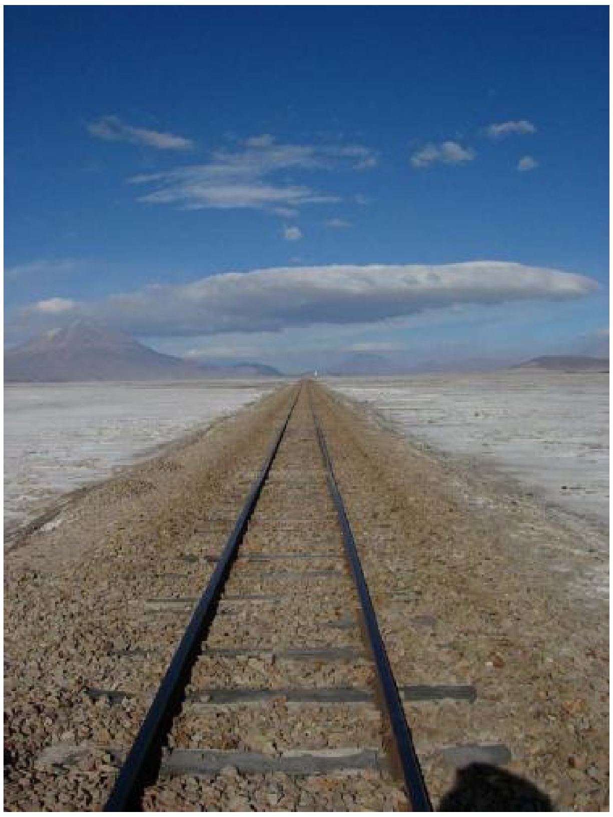


Figure 3.1: *Railway tracks converging at infinity [40]*

3.1 Projective geometry

When talking about projective geometry it is fundamental to introduce a new set of coordinates called *homogeneous*. Any general point $\mathbf{x} = [x, y]^T$ in $\mathbb{R}^2$, written in homogeneous coordinates became $\hat{\mathbf{x}} = [x_1, x_2, x_3]^T$ in the *projective space* $\mathbb{P}^2$. To go back to the inhomogeneous one it's necessary to divide the first two components of $\hat{\mathbf{x}}$ by x_3, in this way $\mathbf{x} = [\frac{x_1}{x_3}, \frac{x_2}{x_3}]^T = [x, y]^T$. As can be immediately noticed, the previous notation holds until $x_3 \neq 0$ such that the homogeneous vector $\hat{\mathbf{x}}$ corresponds to finite points in $\mathbb{R}^2$. The points with $x_3 = 0$ are known as i*ideal* points, or points at *infinity*.

Also lines can be written in homogeneous vector, in this case homogeneous stands for the fact that scaling that particular vector won't change what it represents. A line in the plane is represented by an equation such as $ax + by + c = 0$, different choices of a, b and c giving rise to different lines, thus, a line may naturally be represented by the vector $\mathbf{l} = [a, b, c]^T$. As said before, this is an homogeneous form too since the vector $\hat{\mathbf{l}} = k[a, b, c]^T$ represent the same line as $\mathbf{l}$, for any non zero k.

Here some properties regarding homogeneous points and line will be illustrated. A point $\hat{\mathbf{x}} = [x_1, x_2, x_3]^T$ lies on a line $\mathbf{l} = [a, b, c]^T$ only if $\hat{\mathbf{x}}^T \cdot \mathbf{l} = 0$, this can be easily demonstrated by saying that a point $\mathbf{x} = [x, y]^T = [\frac{x_1}{x_3}, \frac{x_2}{x_3}]^T$ lies on line $\mathbf{l}$ only if $ax + by + c = 0$.
The intersection of two lines $\mathbf{l} = [a, b, c]^T$ and $\mathbf{l}_1 = [a_1, b_1, c_1]^T$ is the point $\mathbf{x} = \mathbf{l} \times \mathbf{l}_1$, From this it's possible to demonstrate that two lines $\mathbf{l} = [a, b, c]^T$ and $\mathbf{l}_2 = [a, b, c_2]^T$ meet indeed at point at infinity, since the vectorial product $\mathbf{l} \times \mathbf{l}_2 = (c - c_2)[b, -a, 0]^T$.

The set of ideal points $[x_1, x_2, 0]^T$ lies on a single line called *line at infinity*, $\mathbf{l}_\infty = [0, 0, 1]^T$

3.1.1 Projective geometry transformation

The transformation in the projective space are very similar to the one of the classic geometry, they are divided in four classes:

- Class I: Isometries
  Isometries are transformations of the plane $\mathbb{R}^2$ that preserve Euclidean distance. *Translation* and *Rotation* belongs to this class and it is described by following equation:

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.1}$$

  where $\mathbf{R}$ is the rotation matrix and $\mathbf{t}$ is the translation vector. If $\mathbf{R}$=$\mathbf{I}$ then the motion is only translational while, if $\mathbf{t}$=$\mathbf{0}$ it is only rotational.

- Class 2: Similarity transformations.
  In this case an isotropic scaling is presents and it follow the relation:

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.2}$$

  where s is the scaling factor.

- Class III: Affine transformation.
  An affine transformation is a non-singular linear transformation followed by a translation. It has the form:

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.3}$$

  where $\mathbf{A}$ is the composition of two fundamentals transformations, namely rotation and non-isotropic scaling.

- Class IV: Projective transformation. This is the most important class in the projective geometry. A *Projective transformation* projects every figure into a projectively equivalent figure, leaving all its projective properties invariant, so it maps points in one $\mathbb{R}^3$ plane to another. It is decribed by:

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^T & v \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.4}$$

  where $\mathbf{v} = [v_1, v_2]^T$. This is an 8 DOF matrix since the only new important parameter is the ratio v. With this transformation parallel lines will not be parallel anymore, instead they will converge at infinity, the ideal point.

All the above discussion as been carried out considering 2D geometric transformation but, 3D ones are not so different, with the only big difference being the definition of the rotation matrix $\mathbf{R}$, which can be represented either through quaternions or euler angles.

3.2 Camera Model

In this section, the simplest model for a camera, which is an object able to map information from a 3D world to a 2D one, will be presented, it is called the *pinhole camera* *model*. Even if this is the most basic model, it is able to adapt to different types of camera, for example X-ray images, scanned photographic negatives, scanned photographs from enlarged negatives, etc.

In this model all the information pass through the camera centre and points are projected into the *image plane* or *focal plane* at a distance f from the centre, a point in space with coordinates $\mathbf{X} = [X, Y, Z]^T$ is mapped to the point on the image plane where a line joining the point $\mathbf{X}$ to the centre of projection meets the image plane. This is shown in Fig. 3.2, and by similar triangle it is possible to quickly compute that the point $\mathbf{X}$ is mapped into the point $x = [f\frac{X}{Z}, f\frac{Y}{Z}]^T$ on the image plane.

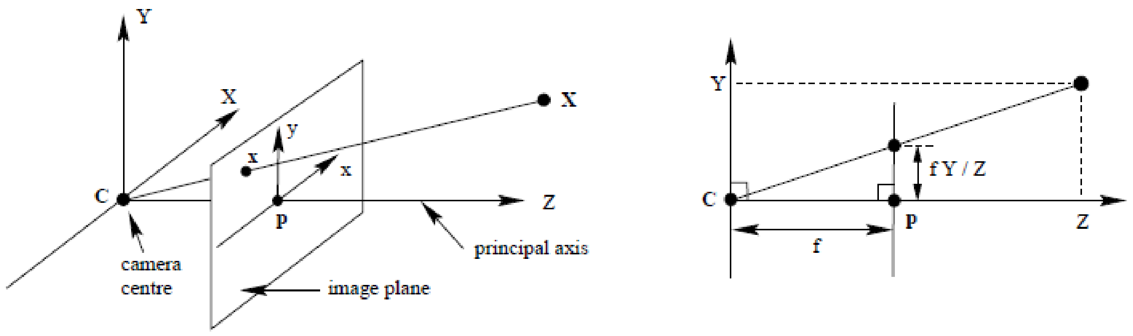


Figure 3.2: *Pinhole camera model [39]*

This is a mapping from an Euclidean $\mathbb{R}^3$ space to an Euclidean $\mathbb{R}^2$ space.

If the world and image points are expressed in homogeneous coordinates, the mapping goes from $\mathbf{X} = [X, Y, Z, 1]^T$ to $\mathbf{x} = [fX, fY, Z]^T$, and can be expressed in matrix form as:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.5}$$

where the 3×4 matrix is called the *camera projection matrix*, $\mathbf{P}$, so that Eq. 3.5 can be written as:

$$\mathbf{x} = \mathbf{PX} \tag{3.6}$$

Until now it has been considered that the origin point of the image plane is at the principal point $\mathbf{p}$, usually this is not the case, so two additional translational DOF for the image plane origin must be added in $\mathbf{P}$, such that:

$$\mathbf{P} = \begin{bmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

where c_x and c_y are the coordinates of the principal point.

In the equations above $\mathbf{X}$ is expressed in the *camera coordinate frame*, in general such a frame is different from the *world coordinate frame*, and the two are related through a rotation and a transaltion. For this reason the notation $\mathbf{X}_{\text{cam}}$ and $\mathbf{X}_{\text{world}}$ will be used, respectively for the point $\mathbf{X}$ in the camera coordinate frame and the world one. The relation between the two points is the following:

$$\mathbf{X}_{\text{cam}} = \begin{bmatrix} \mathbf{R} & -\mathbf{RC} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{X}_{\text{world}} \tag{3.8}$$

where $\mathbf{C}$ represents the coordinates of the camera centre in the world coordinate frame and $\mathbf{R}$ is a 3×3 rotation matrix representing the orientation of the camera coordinate frame. Usually **-RC** is referred as $\mathbf{t}$, as can be seen from Fig. 3.3

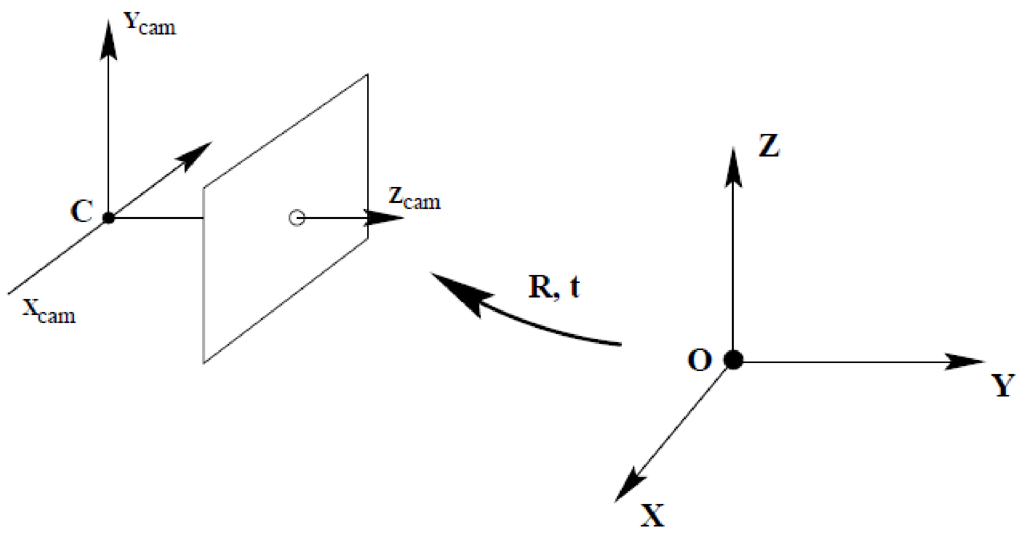


Figure 3.3: *Translation and rotation between the world and the came coordinate frame [39]*

Since the camere and the world coordinate frame are no more coincident, the previous expression of $\mathbf{P}$ is no longer valid. It's new form is:

$$\mathbf{x} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \mathbf{X}_{\text{cam}} = \mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{X}_{\text{world}} = \mathbf{P}\mathbf{X}_{\text{world}} \tag{3.9}$$

$\mathbf{K}$ is called the *camera calibration matrix* and the parameters contained in it are called the *internal* camera parameters, or the *internal orientation* of the camera. The parameters of $\mathbf{R}$ and $\mathbf{t}$ ($\mathbf{C}$) which relate the camera orientation and position to a world coordinate system are called the *external* parameters or the *exterior orientation*. From Eq. 3.9 it is possible to see that the matrix $\mathbf{P}$ has 9 DOF: 3 for K, image axis and focal length, 3 for $\mathbf{R}$ and 3 for $\mathbf{C}$.

It's important that the parameters inside the intrinsic camera matrix **K** have all the same unit, pixels. For this reason the focal length f must be multiplied by the pixel density $[\frac{pixel}{mm}]$, and c_x, c_y can be taken as half resolution of the image in x and y direction.

3.3 Camera calibration

An important step to exploit the full potential of a computer vision software is the Camera Calibration, in fact, an incorrect one would give rise to wrong results, or in the worst case, no results. As always, a model has integrated into it some simplification, indeed the objective of the Calibration step consists in making up the differences between a real camera and the idealized model, modifying, through the use of some derived parameters, the image acquired with a real camera. These parameters are not listed in the camera specification since they are unique to each camera and also differ between cones of the same model or series, and so, they must be calculated.

One of the main reason that this step has to be performed is due to the fact that lenses introduce distortions[41] in the images, to be more specific, two types of it, called *radial distortion* and *tangential distortion* [42]. Truthfully there is another kind of distortion, called *skew*, that represent the skewing of the pixel element in the pixel array such that the image axes x and y are not perpendicular. This is a very rare case and for this reason it will be no further investigated.

Going back to the radial and tangential distortion, as it happens for almost everything, the cheaper the camera, the more apparent they will be.

Starting from the radial distortion, it happens when light rays bend more near the edges of a lens than they do at its optical center, the smaller the lens, the greater the distortion, and consist in seeing straight lines as curved. This effect is shown in Fig. 3.4 and it is analogous to the effect obtained through the use of a fisheye lens. Denoting as x_d and y_d the distorted image points, thei expression is:

$$\begin{cases} x_d = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_d = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{cases} \tag{3.10}$$

where x and y are the coordinates of the undistorted image, as projected by an ideal pinhole camera, the parameters k_i are the radial distortion coefficient and r is defined as:

$$r = \sqrt{(x_d - x_{\text{center}})^2 + (y_d - y_{\text{center}})^2} \tag{3.11}$$

Typically, two coefficients are sufficient for calibration. For severe distortion, such as in wide-angle lenses, 3 coefficients can be selected to include k_3.

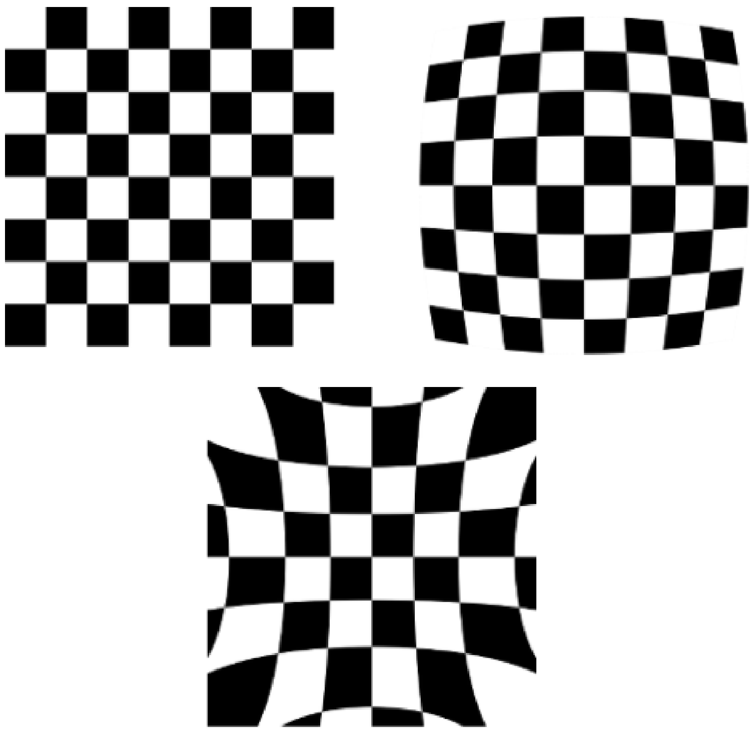


Figure 3.4: *Radial distortion effect: Original image (upper left), Negative radial distortion (upper right), Positive radial distortion(bottom) [42]*

The other distortion, the tangential one, occurs when the lens and the image plane are not parallel, as shown in Fig. 3.5

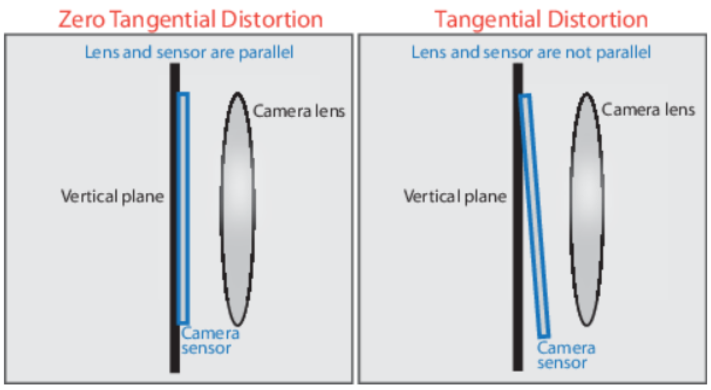


Figure 3.5: *Misalignment of the lens and the image plane [43]*

The effect of the this distortion is reported in Fig. 3.6, it causes some details of the image to be closer, or further apart, than the real case. Similarly to the previous case, it can be expressed as:

$$\begin{cases} x_d = x + [2p_1xy + p_2(r^2 + 2x^2)] \\ y_d = y + [p_1(r^2 + 2y^2) + 2p_2xy] \end{cases} \tag{3.12}$$

The equation above is conceptually equal to Eq. 3.10, with the only difference being the tangential distortion coefficients p_i.

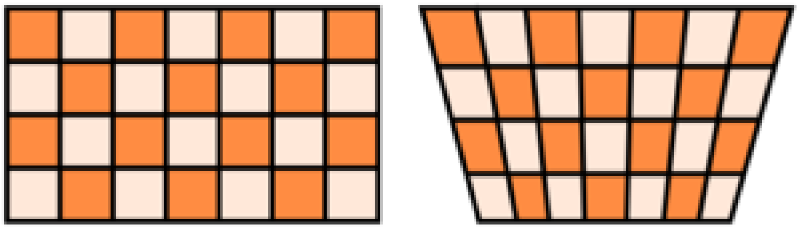


Figure 3.6: *Tangential distortion effect [44]*

These parameters don't change over time so, unless the lenses are changed, or a zoomed is performed (this act on the intrinsic camera parameters), one calibration is sufficient. In [45], the full procedure for calibration is present, here only the fundamental step will be reported.

It starts with the acquisition of a good number of images that have as the subject an object of known pattern, shape and dimensions, usually *checkerboards* are used. These images should be taken all from different angles and the object must be whole in the camera's FOV. Key points are extracted from the images and a comparison is made with those present in the object's pattern. Then a linear mapping between the model points and the 2D image points is constructed, called *Homographies*, $\mathbf{H}$. Then, considering only the simple pinhole projection model, an estimation of $\mathbf{K}$, in closed-form, can be obtained. An unique solution is guaranteed only if at least three images are used, two if the skew is not present. In the previous step no distortions were considered, now, an estimation of its coefficients can be computed using a linear least-squares fitting, minimizing the projection error. The lens distortion parameters are then refined (simultaneously with all other parameters, intrinsic and extrinsic) in a final, overall optimization step, through a non-linear optimization of a function that has as variables all the previous cited parameters.

3.4 Photography and optics

Here, parameters related to a camera and belonging to the fields of photography, optics and the above discussed projective geometry will be described. This is due to the fact that most imaging software use a terminology that is a mixture of terms coming from these fields.

Starting from the lenses, the camera ones can be approximated as converging lenses, shown in Fig. 3.7. that collect the light to the focus.

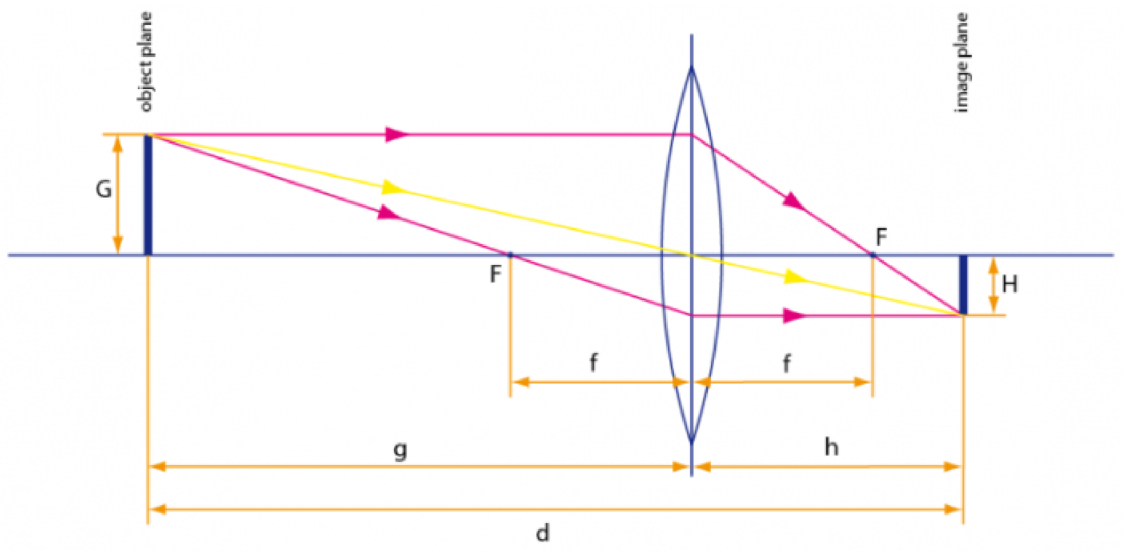


Figure 3.7: *Converging lens [46]*

The equation describing lenses parameters is:

$$\frac{1}{f} = \frac{1}{g} + \frac{1}{h} \tag{3.13}$$

When talking about projective geometry, the focal length f as been identified as the distance between the camera centre and the image plane, but for real cameras this is not valid anymore [47]. The new definition of f is the distance between the lens of the camera and the imaging sensor. The other parameters g and h are, respectively, the distance between the object and the lens, and the distance between the image plane and the lens, G and H are the height of the object and the one of the image. Images that appear upright relative to the object have heights that are positive and those that are inverted have negative heights.

In Fig. 3.9 the effect of the focal length is shown, the shorter it is, the greater the extent of the scene captured by the lens. On the other hand, the longer it is, the smaller the extent captured by the lens. If the same subject is photographed from the same distance, its apparent size will decrease as the focal length gets shorter and increase as the focal length gets longer.

The effect above discussed can be attributed to another parameter, computed starting from the focal length f, called FOV, Field of View. It is the angular extent of a scene that can be captured by the camera and a graphic representation can appreciated in Fig.

3.8. The expression of the FOV is:

$$FOV = 2arctan(\frac{d_{\text{sensor}}}{2f}) \tag{3.14}$$

where d_{sensor} is the size of the imaging sensor, in mm. So from Eq. 3.14 it is possible to better understand the effect discussed before, as the focal length increase, the FOV gets narrower and scene captured is less.

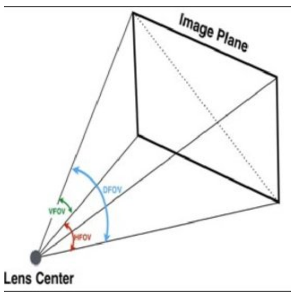


Figure 3.8: *Field of view [48]*

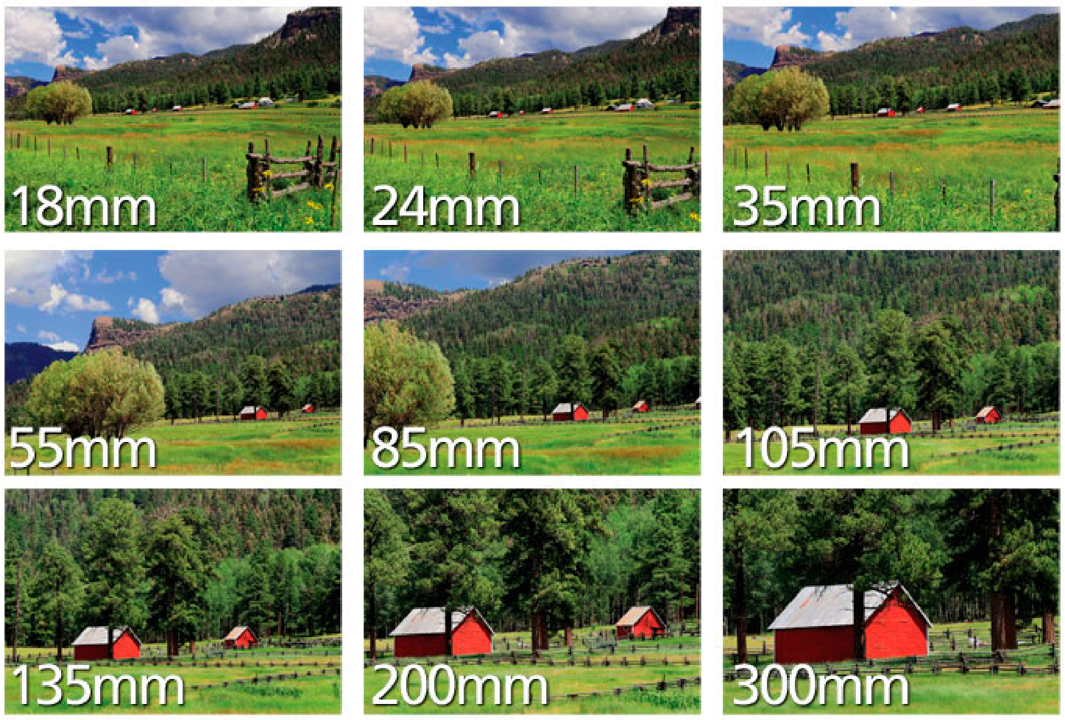


Figure 3.9: *Focal length effect [49]*

Another important parameter is the f-number, it is the measure of how much light enters the camera lens. As for the FOV, it is dependant on f, and another parameter called aperture, a, that represent the size of the lens opening.

$$f_{\text{number}} = \frac{f}{a} \tag{3.15}$$

It could be thought that the bigger the aperture the grater the light that enters the camera. This is relatively true since a bigger aperture will indeed gather more light, however, also the spherical rectangle view of the camera must be taken into account, by means of FOV or f. Just to give an example, a lens with $f = 100[mm]$ and $a = 20[mm]$ will gather as much light as a lens with $f = 50[mm]$ and $a = 10[mm]$, even if the second has half the aperture of the second, due to the fact that their f-number is the same, f/5.

4 Computer Vision

Computer vision is the engineering field that seeks to understand and extract various types of information from images. This chapter contains an insight into this field and an overview of the different methods and techniques found in the literature is presented. Then, a detailed explanation of the techniques used in ORB-SLAM is provided, followed by two view geometry, or epipolar geometry, the method applied to compute the rotation and translation of the camera in time: the fundamental/essential matrix, is estimated from the point tracks and is decomposed into the relative rotation matrix and translation vector. Triangulation allows to compute the initial map, which is later navigated by continuously tracking features and solving the PnP, Perspective n-Point, problem. Finally, the Bundle Adjustment, an optimization method, is described.

4.1 VO and SLAM

As already said in Section 1.2, computer vision algorithms are based on two methods: VO, Visual Odometry, and SLAM, Simultaneous Location And Mapping. The VO estimate progressively the camera pose, taking advantages of the changes present in the images taken, it calculates the transformation that took place between two consecutive images/frames by reconstructing the camera trajectory between the moments of the two snaps. So, in this way, the trajectory is reconstructed incrementally, image after image.

The SLAM problem was first introduced in the field of robotics. As the name implies, it is a process by which it is possible to reconstruct simultaneously the surrounding environment and determine the position of the robots, equipped with a camera, in it. This happens through the observation of landmarks, points of the surrounding environment that are taken as a reference. The reconstruction of the environment and the determination of the position are estimated through a probabilistic algorithm. Since the position depends on the quality of the reconstructed map and the relative measurements of the sensor, the better the map was reconstructed, the better the accuracy of the determined position. As mentioned, the consistency of the trajectory, local for VO and global for SLAM, is the difference between the two mehods, since in the second the *loop closure* is added. With it, it is possible to understand if a place has already been visited or not and, taking advantage of this information, reduce the drift accumulated over time. In fact, if not counterbalanced by the readings of other sensors (IMU,GPS,et.), the drift presents in the reconstructed trajectory will continue to grow, leading, in the long run, to a significant error. In Fig. 4.1 a simple example is illustrated. The circle (A) is the ground truth while (B) is the reconstructed trajecotry with the presence of drift and the absence of a loop closure. The point P1 should coicide with the point P2 but, due to the drift, this doesn't happen. Using the VO method, case (A) would be be obtained only in ideal condition, by not considering drift, or, using other sensors' information to make correction. However, even if these ideal conditions are considered, the points P1 and P2 will always be seen as separate, even if overlapping. Instead, using SLAM, the loop closure forces the overlapping of the two points by making them coincide in P, it also corrects the entire trajectory thus obtaining the result in (A). Thanks to the loop closure, SLAM manages, as mentioned, to correct the entire list of stored landmarks and therefore to improve the accuracy of the results. All of these better results come with a price, although better, SLAM is also more computationally expensive.

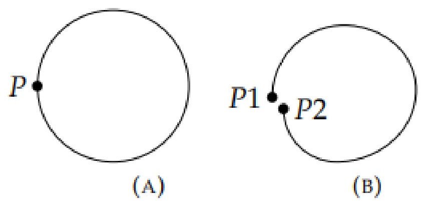


Figure 4.1: *Loop closure example*

4.2 Direct and Feature-based methods

The methods that can be used to reconstruct the position of the camera and the environment from the captured images are two, both for VO and SLAM:

- *Direct method*: the intensity value in the entire image is used as base for the estimation, the difference in the intensity of the pixels among those observed is used for the relative position reconstruction of the frame taken into consideration.

- *Feature-based method*: differently from the previous method, it is no longer based on the whole image, but only on some elements, the *features*. Once identified in an image, they are looked for and recognized in the following images by the descriptors. So, using camera models, like the pinhole camera one described in Section 3.2, they transform information from the 2D to the 3D world by obtaining the camera position.

The first method is less accurate, slower and computationally more expensive than the second one. For these reasons the feature-based methods are further investigated in the following sections.

4.2.1 Feature detection

Features are specific points in the image that can be easily identified, like angles, sharp lines, or high contrast areas. Feature detection is therefore a fundamental step, nevertheless, the first one to be done, since it retrieves information from the image.

In literature it's possible to find several methods that allow the extraction of features, with different properties between each other. The *Harris detector* is the most classic and intuitive one, it evaluates the change in intensity if a pixel with respect to the surrounding area, if it is high then a feature is detected. Among the other methods there are FAST,SURF, BRIEF and the one used in this thesis, ORB [50]. Two of them will be rapidly analyzed, Features from Accelerated Segment Test select and understand if a pixel is of interest or not. It analyze both the intensity of the pixel and those around it, using an appropriate threshold it decide whatever the examined pixel is a corner or not. the Speed Up Robust Features, SURF, obtains the feature thanks to the use of an approximation of the Hessian matrix and exploiting the so-called integral images. Taking x as the location, the integral image corresponds to the sum of the pixels of a rectangular area formed by the point x and the origin.

4.2.2 Feature Matching and Tracking

After extracting good features from an image, the next step is the one of finding the in the following frames/images. Two methods are commonly used:

- **Feature Matching**: This method is based on descriptors, they are extracted for each detected feature in the image and then, are compared with the one of the subsequent image.The one that are more similar will get matched and a track will be created. The results are not always accurate and false tracks could be present, due to the fact that points could be similar to each other, so they must be removed. A popular algorithm used to remove false track is RANSAC [51], RANdom Sample Consensus. It is a simple and efficient iterative algorithm for outliers detectors. It is accomplished in the following step: *1)Randomly selecting a subset of the data set, 2)Fitting a model to the selected subset, 3)Determining the number of outliers, 4)Repeating steps 1-3 for a prescribed number of iterations.* Then the model with the most number of inliers is selected between all the computed ones. RANSAC is already implemented in Matlab [52].

- **Feature Tracking**: This method extract features only in the first image and they are tracked predicting their future position in the following images. At this point, in the subsequent image, each initial features is searched in a neighbourhood of the predicted point. In literature it is possible to find several tracker, each with its own characteristics.

4.3 ORB-SLAM

In this thesis the visual SLAM approach of ORB-SLAM is exploited. To the SLAM methodology, the modified version of FAST and BRIEF for features detection and description is added. In order to obtain information for reconstructing the camera pose feature matching is used.

4.3.1 General SLAM algorithm

The SLAM algorithm is based on the solution of the localization problem, in particular, on the minimization of the uncertainties introduced during the movement of the robot [53]. Defining:

- x_k Pose of the robot.
- $l = l_1, l_2, ..., l_n$ Set of all landmark position.
- $Z = z_1, z_2, ..., z_k$ Set of all landmark observations.
- $U = u_1, u_2, ..., u_k$ Control input history.
- x_0 Initial pose.

The distribution probability can be expressed as:

$$P(x_k, l|Z, U, x_0) \tag{4.1}$$

Since it's not possible to directly found the robot's pose and the set of all landmarks, the SLAM algorithm tries to find what they could be. So, the estimation of the pose and the map depends on two factors: the control input and the observations. The former derives from the robot's internal sensor and is used for a first estimate:

$$P(x_k|x_{\text{k-1}}, u_k) \tag{4.2}$$

where $x_{\text{k-1}}$ is the last pose and u_k is the current control input. The observation factor is due to the optical sensors of the robot:

$$P(z_k|x_k, l) \tag{4.3}$$

The problem presented in 4.1 can now be solved, initially the *Time-update* is required:

$$P(x_k, l|Z_{\text{0:k-1}}, U_{\text{0:k}}, x_0) = \int P(x_k|x_{\text{k-1}}, u_k) \cdot P(x_{\text{k-1}}, l|Z_{\text{0:k-1}}, U_{\text{0:k-1}}, x_0)dx \tag{4.4}$$

Then, the *Measurement-update* will be calculated:

$$P(x_k, l|Z_{\text{0:k}}, U_{\text{0:k}}, x_0) = \frac{P(z_k|x_k, l)P(x_{\text{k}}, l|Z_{\text{0:k-1}}, U_{\text{0:k}}, x_0)}{P(z_k|Z_{\text{0:k-1}}, U_{\text{0:k}})} \tag{4.5}$$

4.3.2 Oriented FAST and Rotated BRIEF

Oriented FAST and Rotated BRIEF (ORB) is the features detection and description method used in ORB-SLAM, and, as can be deduced from its name, it is based on two other features extractors and descriptors, FAST and BRIEF. The former was already discussed in one of the previous sections, while the latter has not. BRIEF is a descriptor that uses binary strings to describe image patched, it creates short strings directly after comparing the pairs of points and their intensity. ORB uses variant both for FAST and BRIEF, the first is called oFAST, that after performing FAST operations organizes, among the extracted features, the best ones in a multi-scal pyramid scheme. It also take into consideration their orientation component. The second is called rBRIEF and orientation operations are also performed here, in particular, BRIEF is oriented along the direction of the keypoints through rotation matrices.

With this procedure a process-friendly property is lost, the high variance linked to each bit feature. To put a limit to it, after orienting BRIEF, a selection is made, keeping only those points with an high variance and a mean close to 0.5. With this, ORB turns out to be one of best algorithm available, both in terms of speed and computational cost.

4.4 Epipolar geometry

The epipolar geometry is the intrinsic projective geometry between two views. It is independent of scene structure, an only depend on the camera internal parameters and relative pose. The geometric entities involved in epipolar geometry are illustrated in Fig. 4.2 and they are:

- The **epipoles, e** and **e'**, are the points of intersection of the baseline, line joining the camera center **C** and **C'** with the image plane, or, equivalently, the image in one view of the camera centre of the other view.
- The **epipolar plane** is the plane containing the baseline.
- The **epipolar line** is the intersection of an epipolar plane with the image plane. All epipolar line intersect at the epipole.

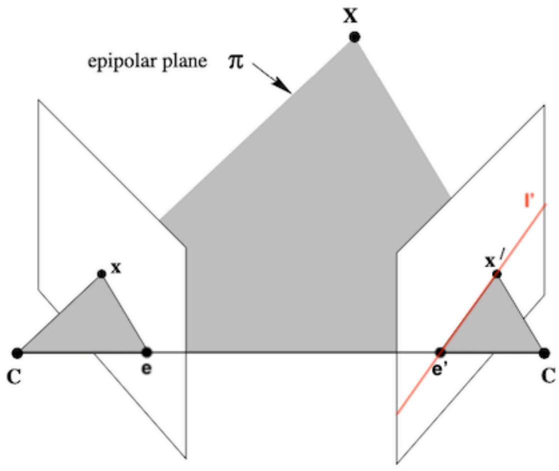


Figure 4.2: *Two view geometry*

Given a point **X** in a 3D space, it is imaged in two views, **x** in the first, and **x'** in the second. The relations between the three points, and the rays back-projected from **x** and **x'** that intersect at **X**, is that they are coplanar in the plane $\boldsymbol{\pi}$. Supposing that only **x** is known it is possible to obtain **x'** due to some constrain. The plane $\boldsymbol{\pi}$ is determined by the baseline and the ray defined by **x**. From before, it is known that the ray corresponding to the point in **x'** lies in $\boldsymbol{\pi}$, hence the point **x'** lies on the line of intersection **l'** of $\boldsymbol{\pi}$ with the second image plane. This line is the image in the second view of the ray back-projected from **x**, also called *epipolar line.*
This constrain can be expressed in a mathematical formulation through the essential matrix **E**, which maps points in one image to the corresponding epipolar lines in the othe one:

$$\mathbf{E}\hat{\mathbf{x}}^{T} = \mathbf{l'} \tag{4.6}$$

where $\hat{\mathbf{x}} = \mathbf{K}^{-1}\mathbf{x}$ notation refers to normalized coordinates, where the camera matrix **P** becomes **I[R|t]**. The epipolar constrain is described as:

$$\hat{\mathbf{x}}'_{\mathrm{T}}\mathbf{E}\hat{\mathbf{x}} = 0 \tag{4.7}$$

Through a series of geometric properties, the matrix **E** is defined as:

$$\mathbf{E} = [\mathbf{t}]_x \mathbf{R} \tag{4.8}$$

Where **R** is the rotation matrix from camera frame two to one and **t** is the translation vector of frame two written in frame 1. This equation can be used to recover analytically the expression of **E** given the two camera poses as well as to retrieve the extrinsic parameters of the relative camera pose if an estimate of **E** is available.

To calculate **E**, a priori knowledge of **K** is necessary, but, when it is not available, a genralization of the essential matrix can be used, the fundamental matrix **F**. The relation between the two is the following:

$$\mathbf{E} = \mathbf{K}^T \mathbf{F} \mathbf{K} \tag{4.9}$$

So, as can be deduced from the fact that there are more unknowns, **F** has more degree of freedom, 7, with respect to **E**, 5. The considerations done before still holds for **F**.

4.5 Fundamental and Essential matrix estimation

Given a set of point correspondences in two images, **F** and **E** can be estimated from them. Due to the previously cited fact that **E** has a lower number of degrees of freedom it can also be reconstructed from a lower number of correspondences, with respect to **F**, even though the complexity of the algorithm is increased.

The 8 points normalizes algorithm is the simplest one for the estimation of **F**. By recalling the epipolar constrain, $\mathbf{x'}^{\mathrm{T}}\mathbf{F}\mathbf{x}=0$, one linear equation in **F** components can be obtained per points pair. Once obtained, the set of equations can be rearranged in a linear systems giving the following result:

$$\mathbf{Af} = \begin{bmatrix} x'_1 x_1 & x'_1 y_1 & x'_1 & y'_1 x_1 & y'_1 y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ x'_n x_n & x'_n y_n & x'_n & y'_n x_n & y'_n y_n & y'_n & x_n & y_n & 1 \end{bmatrix} \mathbf{f} = 0 \tag{4.10}$$

where **f** is the vector containing the component of **F**:

$$\mathbf{f} = [F_{11}, F_{12}, F_{13}, F_{21}, F_{22}, F_{23}, F_{31}, F_{32}, F_{33}]^T \tag{4.11}$$

and **n** is the index for the n-th correspondence. Unfortunately, during feature matching or tracking points couple could be not exact, due to noise or some other interference, and in this case a least square solution must be sought for that system.

4.6 Extrinsic parameters from essential matrix

As said before, if an estimation of $\mathbf{E}$ is available, it is possible to extract the relative extrinsic parameters between two different camera poses. Since the essential matrix is an homogeneous one, the camera matrix can be reconstructed up to scale. It is possible to demonstrate that, for a SVD up to scale of $\mathbf{E}{=}\mathbf{U}\,diag(1{,}1{,}0)\mathbf{V}^{\mathrm{T}}$ and $\mathbf{P}{=}\ [\mathbf{I}|\mathbf{0}]$, four possible choices for $\mathbf{P'}$ are available.

Since by definition $[\mathbf{t}]_{\mathrm{x}}\mathbf{t}{=}\ \mathbf{0}$, it follows that $\mathbf{t}{=}\mathbf{U}(0{,}0{,}1)^{\mathrm{T}}{=}\mathbf{u}_3$, the last column of $\mathbf{U}$, but since is up to scale the sign is not known.$[\mathbf{t}]_{\mathrm{x}}$ may be written as $[\mathbf{t}]_{\mathrm{x}}{=}\mathbf{UZU}^{\mathrm{T}}$ and it can be demonstrated that it can also be decomposed as $\mathbf{U}\,diag(1{,}1{,}0)\mathbf{WU}^{\mathrm{T}}$ or equally $\mathbf{U}\,diag(1{,}1{,}0)\mathbf{W}^{\mathrm{T}}\mathbf{U}^{\mathrm{T}}$ with $\mathbf{Z}{=}diag(1{,}1{,}0)\mathbf{W}$ or $\mathbf{Z}{=}diag(1{,}1{,}0)\mathbf{W}^{\mathrm{T}}$.

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.12}$$

$$\mathbf{Z} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.13}$$

Finally, since $\mathbf{E}{=}[\mathbf{t}]_{\mathrm{x}}\mathbf{R}{=}\mathbf{UZU}^{\mathrm{T}}$, the expressions of $\mathbf{R}$ can be:

$$\mathbf{R} = \mathbf{UWV}^T \quad or \quad \mathbf{R} = \mathbf{UW}^T\mathbf{V}^T \tag{4.14}$$

Since the scale of $\mathbf{E}$ is not defined, the sign is not set too and both $\mathbf{R}$ expressions are valid. With two possibilities fot the translation vector and two for the rotation matrix, a total of four solutions are possible, shown in Fig. 4.3

The correct solution is the one where the point $\mathbf{X}$ is in front of both cameras.

4.7 Triangulation

Given two image points $\mathbf{x}$ and $\mathbf{x'}$ and their respective camera matrices $\mathbf{P}$ and $\mathbf{P'}$, the method by which the world point $\mathbf{X}$ is found is the *triangulation* one. The easiest methods for triangulation are the linear ones, the system of equation is obtained by expanding the expression $\mathbf{x}{=}\mathbf{PX}$ and $\mathbf{x'}{=}\mathbf{P'X}$ and is defined by the form $\mathbf{AX}{=}0$ with $\mathbf{A}$:

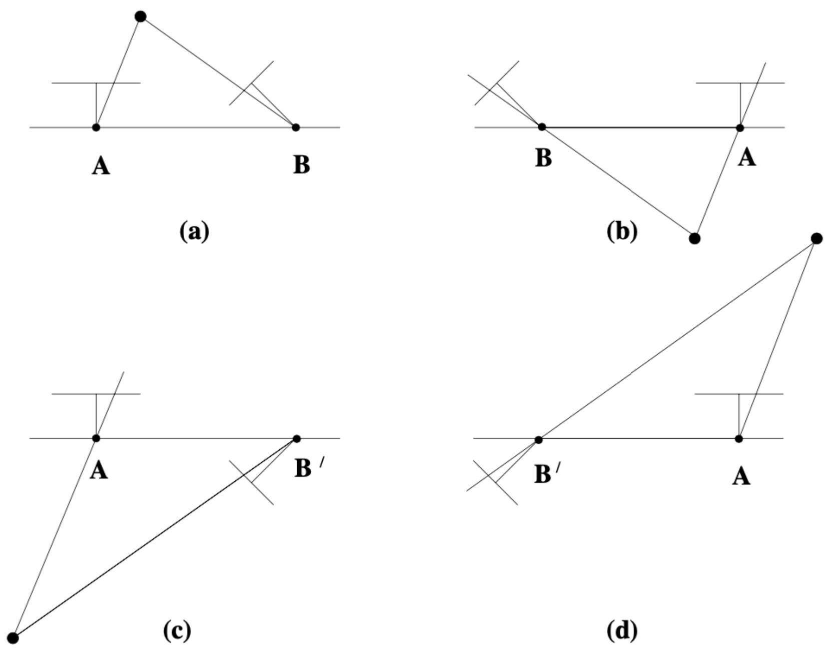


Figure 4.3: *The four possible solutions for calibrated reconstruction from* $\boldsymbol{E}$

$$\mathbf{A} = \begin{bmatrix} x\mathbf{P}_3^T - \mathbf{P}_1^T \\ y\mathbf{P}_3^T - \mathbf{P}_2^T \\ x'\mathbf{P'}_3^T - \mathbf{P'}_1^T \\ y'\mathbf{P'}_3^T - \mathbf{P'}_2^T \end{bmatrix} \tag{4.15}$$

where the notation $\mathbf{P}_n$ refers to the rows of $\mathbf{P}$. When the system is solved through SVD, the method is referred as DLT. The solution is the singular vector corresponding to the singulare value of $\mathbf{A}$.

The method just described consider images, and so image points, as perfect, without any kind of noise/distortion, but in real application this is not true, geometrical relations are no longer exactly satisfied and optimal algorithms will be needed.

Since a more detailed description of the topic of epipolar geometry is out of the scope of this work, if interested, the author invites the reader to consult [39].

4.8 Perspective n-Point Problem

Perspective-n-Point is the problem of estimating the pose of a calibrated camera given a set of n 3D points in the world and their corresponding 2D projections in the image. The camera pose consists of 6 DOF which are made up of the rotation (roll, pitch, and yaw) and 3D translation of the camera with respect to the world. A commonly used solution to the problem exists for n = 3 called P3P, and many solutions are available for the general case of n $\geq$ 3. A solution for n = 2 exists if feature orientations are available at the two points. Some methods, such as UPnP [54] or the Direct Linear Transform, DLT, applied to the projection model, don't consider the intrinsic parameters as already known and so they try to estimate them as well as the extrinsic parameters which make up the pose of the camera that the original PnP problem is trying to find. RANSAC is also commonly used with a PnP method to make the solution robust to outliers in the set of point correspondences.

The simplest form of the PnP problem can be retrieved by expanding the relations **x**=**PX**:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \mathbf{K} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} \tag{4.16}$$

and by passing from homogeneous coordinates to normal ones, $\hat{x} = x/w$ and $\hat{y} = y/w$, and creating a vector with the m_{ij}, it is possible to create a linear system:

$$\begin{pmatrix} X & Y & Z & 1 & 0 & 0 & 0 & 0 & -\hat{x}X & -\hat{x}Y & -\hat{x}Z & -\hat{x} \\ 0 & 0 & 0 & 0 & X & Y & Z & 1 & -\hat{y}X & -\hat{y}Y & -\hat{y}Z & -\hat{y} \end{pmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ \vdots \\ m_{34} \end{bmatrix} \tag{4.17}$$

With at least $n > 6$ correspondences the linear system $\mathbf{Am} = 0$ can be solved through the SVD method [55].

Talking about the previously mentioned P3P problem, it is described by the following linear system:

$$\begin{cases} Y^2 + Z^2 - YZp - b^2 = 0 \\ Z^2 + X^2 - XZq - c^2 = 0 \\ X^2 + Y^2 - XYr - a^2 = 0 \end{cases} \tag{4.18}$$

This formulation is provided in Matlab and is joined with RANSAC too for outliar removal.

When $n \geq 4$ another formulation can be adopted, called EPnp, Efficieent PnP. If interested, its formulation can be found in [56]

4.9 Bundle Adjustment

Bundle adjustment is an optimization technique generally found at the last steps, at least its global version, of a computer vision algorithm. It is tasked with the refining of the model of the structure of the scene, the motion of the camera in terms of relative pose and 3D points. The equation $\mathbf{x}$=$\mathbf{PX}$, that project a 3D world point $\mathbf{X}$ into a 2D point $\mathbf{x}$ by using the camera matrix $\mathbf{P}$, in presence of noise is never solved exactly. For this reason, estimated quantities must be used such that:

$$\hat{\mathbf{x}} = \hat{\mathbf{P}}\hat{\mathbf{X}} \tag{4.19}$$

Making the assumption of a Gaussian noise, the goal is to estimate the three parameters above such that the equation can be solved exactly, in this way, the reprojection error between the estimated 2D point, $\hat{\mathbf{x}}$, and the real one, $\mathbf{x}$, is minimized. This procedure must be performed for every 3D world point and for every available camera pose:

$$\hat{\mathbf{x}}_j^i = \hat{\mathbf{P}}^i\hat{\mathbf{X}}_j \tag{4.20}$$

where the index i stands for the i-th camera view and j for the j-th point. The equation for the reprojection error is written as:

$$\min_{\hat{\mathbf{P}}^i, \hat{\mathbf{X}}_j} \sum_{i,j} d(\hat{\mathbf{P}}^i\hat{\mathbf{X}}_j, \hat{\mathbf{x}}_j^i) \tag{4.21}$$

with d describing the geometric distance between the estimated and real 2D points. Since the number of parameters involved is quite high, the global bundle adjustment quickly become a huge minimization problem requiring an important processing power, which is typically not available in a space mission scenario. Various solutions to this problem exist, starting from, maybe, the most intuitive ones, that consist on applying the method only to a moving window between frames or to carefully selected frames, also known as keyframes. Other solutions regard the possibility of applying the bundle adjustment method selectively, to optimize structure only, the 3d points in the scene, or motion only, the camera parameters, trajectory included, to provide further customization options.

5 Thesis workflow and set-up steps

In this section the thesis workflow will be shown, starting from the target identification and finishing the V-SLAM algorithm used. In Fig. 5.1 its diagram representation can be appreciated.

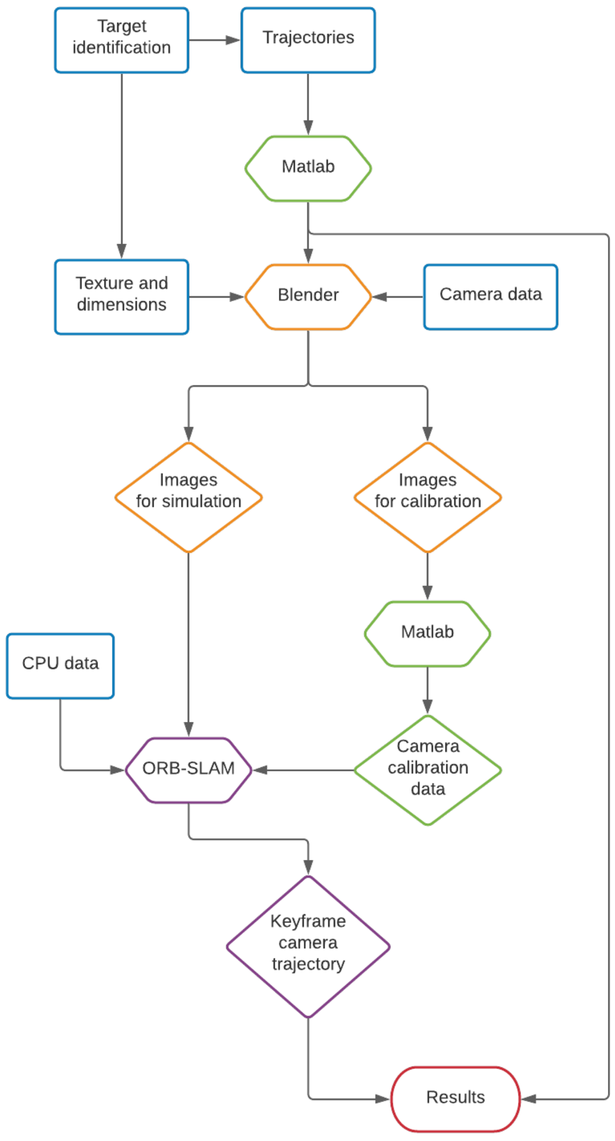


Figure 5.1: *Thesis workflow*

The rectangular blocks represent the external data that must be obtained, in other words, the input, the hexagonal ones are the programs used to analyse and process the various data, a color is given to each program and to the subsequent data, the rhombus are these data and the rounded shape represents the final results obtained.

As can be seen, everything starts from the target identification, once one has been selected and all the necessary data are obtained it is possible to progress to the next step, the creation of synthetic images. After the rendering of the chosen relative trajectories is done and so the frames of its motion are available, they are fed to the ORB-SLAM algorithm together with the data obtained from the camera calibration and, if requested, the computational power is suppressed in order to have a more realistic model, simulating also the relatively low computational power typical of space missions. Then, all the useful data going out of the ORB-SLAM algorithm are taken, the significant ones are the camera trajectory and the time used to process all the data, and are discussed in the results section, also comparing the reconstructed trajectory to the real one. In the following section the target chosen is presented, along with all its parameters and specifications.

5.1 VESPA

The space object chosen as target for this thesis is a space debris called VESPA, VEga Secondary Payload Adapter, to be more specific, its upper part. It is an adapter of the VEGA launcher injected in orbit in 2013 during the launch of Proba-V spacecraft, it can be seen in Fig. 5.2. VESPA is a challenging object for a visual system as it is almost completely black and is an axisymmetric object. Moreover it is an important target for European space debris removal missions under the ESA project ADRIOS, Active Debris Removal/ In-Orbit Servicing.

5.1.1 VESPA's physical characteristics

VESPA is composed by an upper part and a lower part, attached by means of a clamp-band with low shock separation system. Proba V was connected to VESPA by means of another clamp-band with low shock separation system. Its physical characteristics are the following [58]:

- Total height: 4.552 m
- Maximum diameter: 2.62 m
- Total mass: 455 kg
- Materials CFRP and alluminum alloy

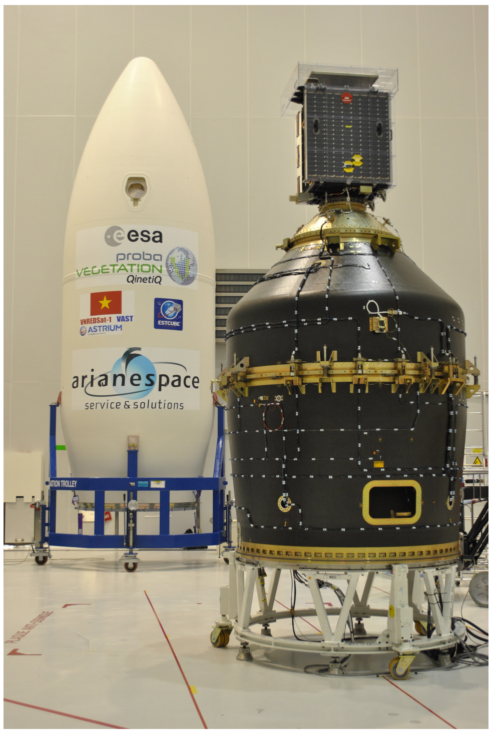


Figure 5.2: *VESPA with Proba-V on top [57]*

As mentioned before the interest is focused on VESPA's upper pat, so, considering only that, the debris characteristic dimension should be of about 2 m, coherently with the radar cross-section of 3.911m2. The most important factors to be considered for the inspection imaging strategy, and for the creation of the model in blender, are the following ones:

- The target is mainly composed by a low reflection material (black), the CFRP.
- There are some parts, i.e. the clamp bands, made by high reflection materials.
- The target has an axial symmetric shape.

In Fig. 5.3 a closeup of VESPA upper part can be seen and some interesting elements/features can be appreciated. This kind of analysis in performed in order to collect useful data, on one side for the model to be created on blender, on the other, to understand the required images resolution at the closest inspection. A list containing some of the potentially interesting elements is reported in Tab. 5.1

It must be remembered that these are not all of the interesting elements that can be seen on VESPA, also bolts and all the other polished, so with high reflection coefficient, components should be taken into consideration.

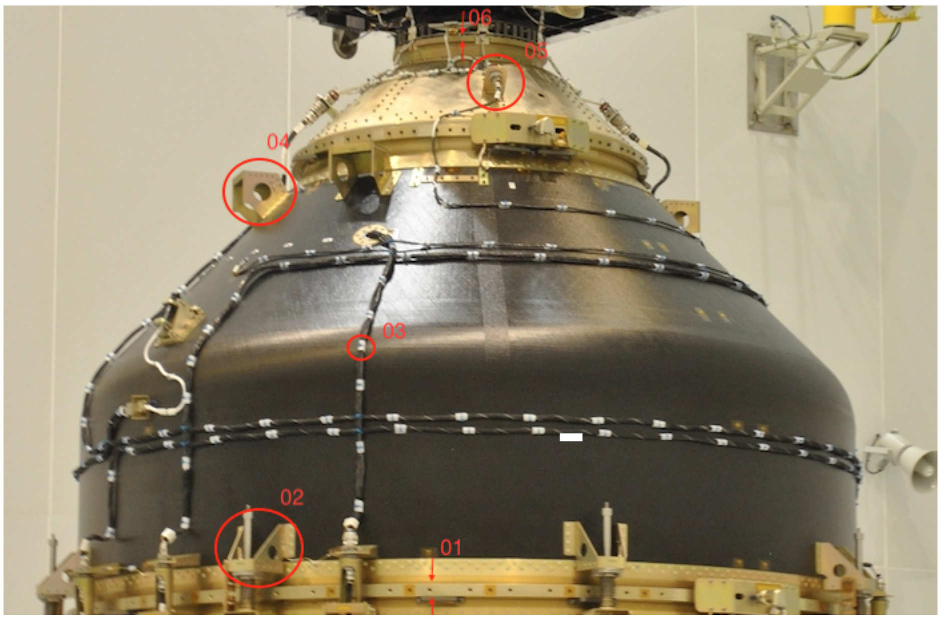


Figure 5.3: *Interesting elements/features of VESPA upper part*

Element ID	Description	Material
01	lower flange	polished metal
02	bracket below upper ring	polished metal
03	cables holder	metal
04	-	polished metal
05	supporting bracket for spring bolts	polished metal
06	upper flange	polished metal

Table 5.1: *VESPA possible elements of interest*

5.1.2 VESPA's orbit

Regarding it's orbit around the Earth, the orbital elements, available in the NORAD catalogue [59] are reported in the following Tab. 5.2

Orbital element	Value
Semi-major axis [km]	7109.7962
Eccentricity [-]	0.009518
Inclination [deg]	98.8425
RAAN [deg]	324.1226
Arg. of pericenter [deg]	337.8519

Table 5.2: *VESPA's orbital elements*

In Fig. 5.4 its orbit trajectory in the ECI reference frame is presented, while, the orbit trajectory in the SCI reference frame is shown in Fig. 5.5.

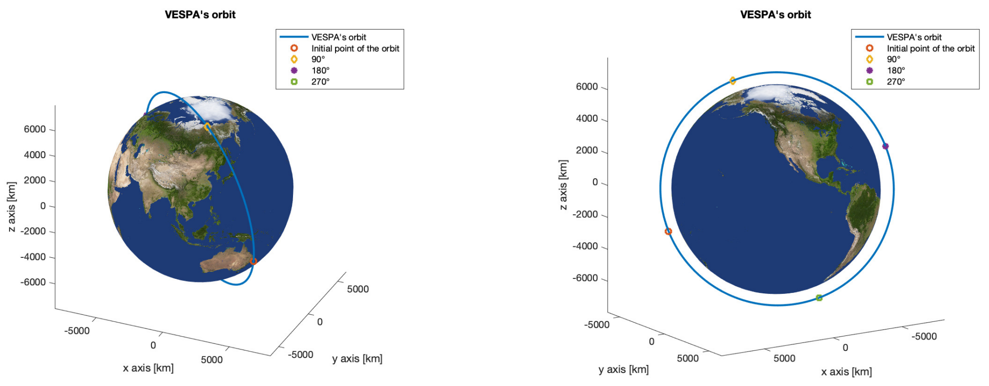


Figure 5.4: *VESPA's orbit in the ECI reference frame*

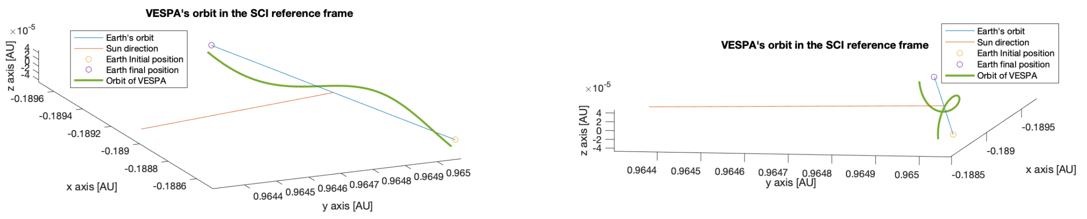


Figure 5.5: *VESPA's orbit in the SCI reference frame*

5.2 Creation of synthetic images in Blender

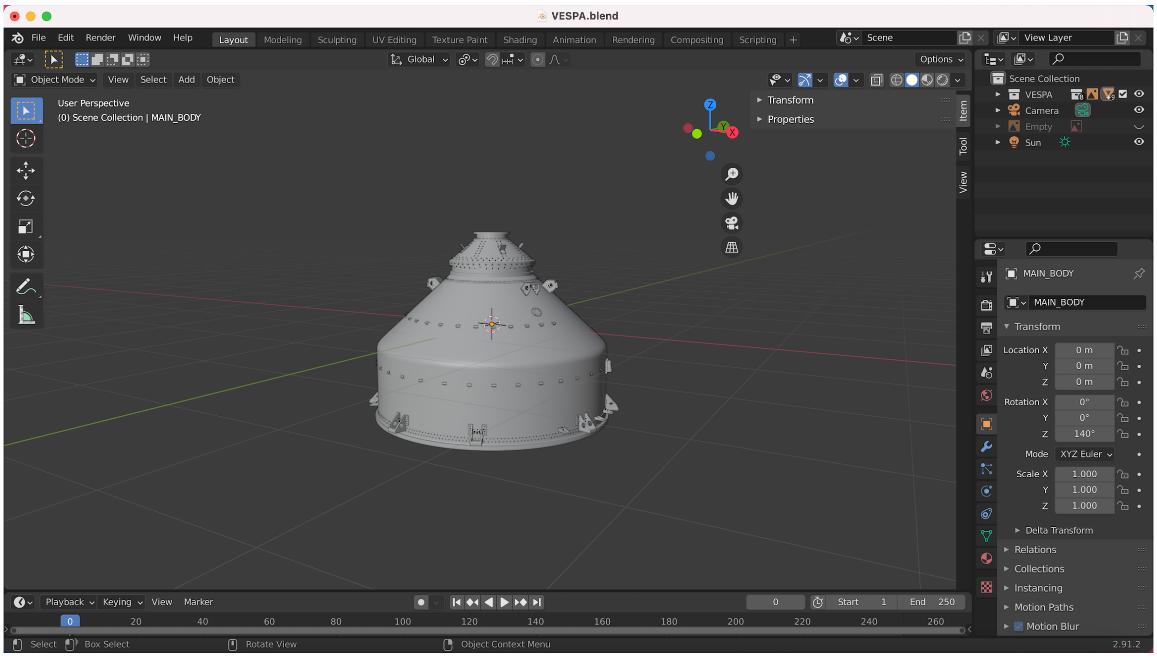
The creation of synthetic images is one of the main stages of this work. It puts the basis for which the results are obtained, its outputs in fact are the generation of synthetic images for the algorithm of recognition and tracking of the features and the images for the calibration algorithm of the camera. The former are generated with the aim of creating a scenario that is as realistic as possible, accordingly to some criteria that will be lately explained. The other objective is no longer photorealism, but the precise reconstruction of the calibration grid and the accurate simulation of the step that should be performed in the case of calibration of a real camera, this specific argument will be discussed in Section 5.3.

Regarding the criteria mentioned before, they are the rendering time and the graphic rendering, or photorealism. The first is due to the fact that each trajectory (animation) is composed of many frames and the time to render each frame should be reasonable, not taking an enormous amount of time, and computational power. The second, as the name implies, is due to the photorealism of the scenario, i.e. an image with a grainy texture is not beautiful to see and doesn't describe the reality in an accurate way, or also, the correct scale should be used between the S/C and the space environment, according also to the software capabilities.

As mentioned in Section 1.3, the software utilized for the creation of synthetic images is Blender, to be more precise, the version 2.91 of it. Numerous tutorials can be found online, one of the most basic, that covers all the principle fields is [60].

5.2.1 VESPA's model

Once the target is defined and all its dimensions are acquired, or at least the main ones, it is time to pass on the creation of the model in Blender. This step is very similar to the creation of a model in a CAD software, in fact blender sustain this kind of feature. In Fig. 5.6 the model in the blender environment is shown, the main body and all of the interesting elements can be appreciated.


VESPA.blend
File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compositing Scripting
Scene
View Layer
Global
Options
Object Mode View Select Add Object
User Perspective
(0) Scene Collection | MAIN_BODY
Transform
Properties
Item
Tool
View
Scene Collection
VESPA
Camera
Empty
Sun
MAIN_BODY
MAIN_BODY
Transform
Delta Transform
Relations
Collections
Instancing
Motion Paths
Motion Blur
Playback Keying View Marker
Select Box Select Rotate View Object Context Menu
2.91.2

Location X 0 m
Y 0 m
Z 0 m
Rotation X 0°
Y 0°
Z 140°
Mode XYZ Euler
Scale X 1.000
Y 1.000
Z 1.000

0 Start 1 End 250

Figure 5.6: *CAD model of VESPA*

Once the model is deemed satisfactory, it is time to pass onto the next step, the creation of the materials for it, that can be simplified into the application of a texture to the selected part of the model. As can be seen from the various images in the previous section, and it also written, VESPA is composed mainly by a low reflection material, CFRP, and a high reflective one. The latter can be divided into two different types of material, with the only difference being in the color, the first one, that can be seen almost everywhere, is very similar to the "color" of gold, while the second, a bit more hidden, is more like the "color" of aluminum, and can be found in the bolts of the upper part of the target and in the cables holders. To create these three materials the textures present in Fig. 5.7 were used.

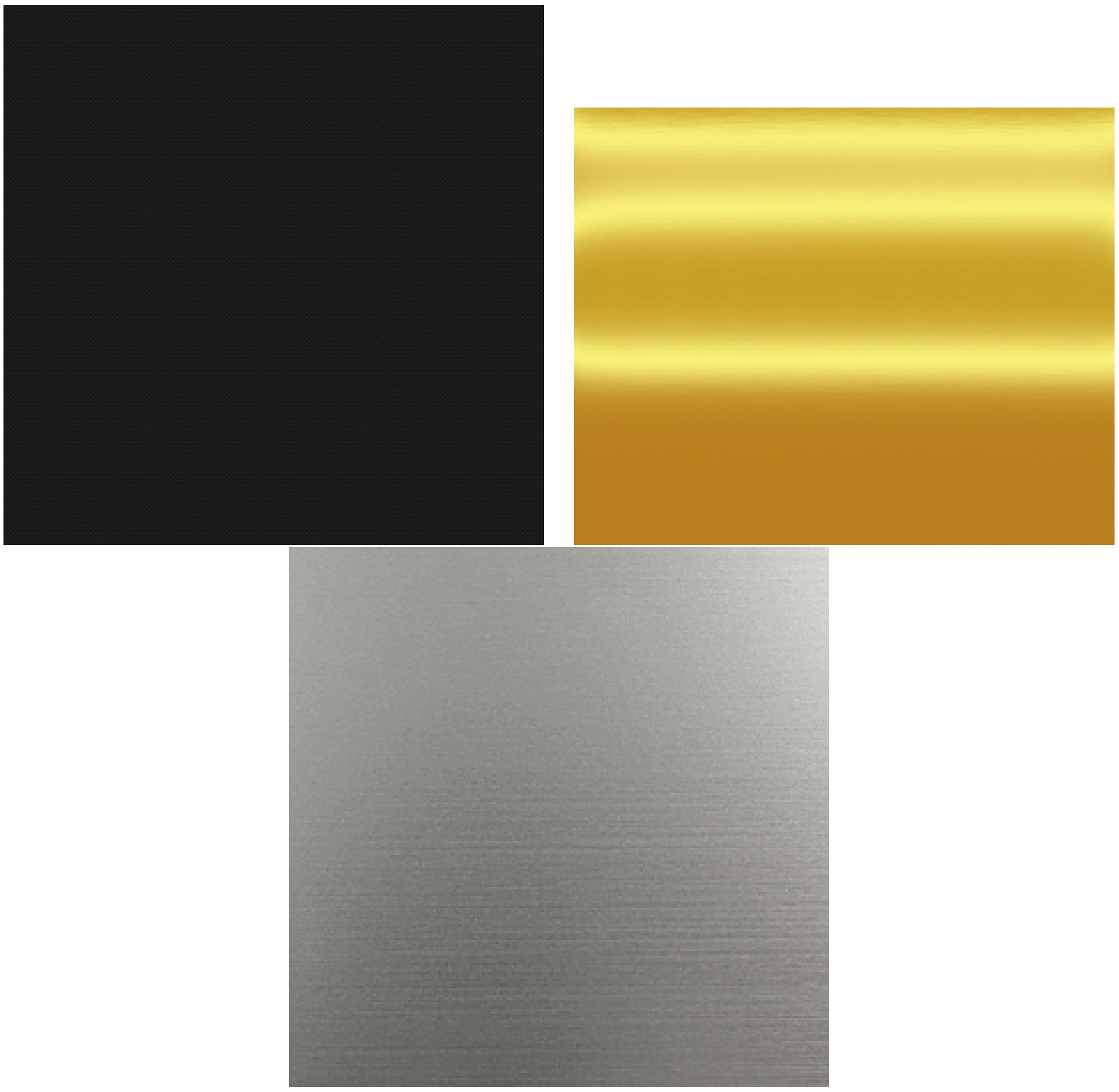
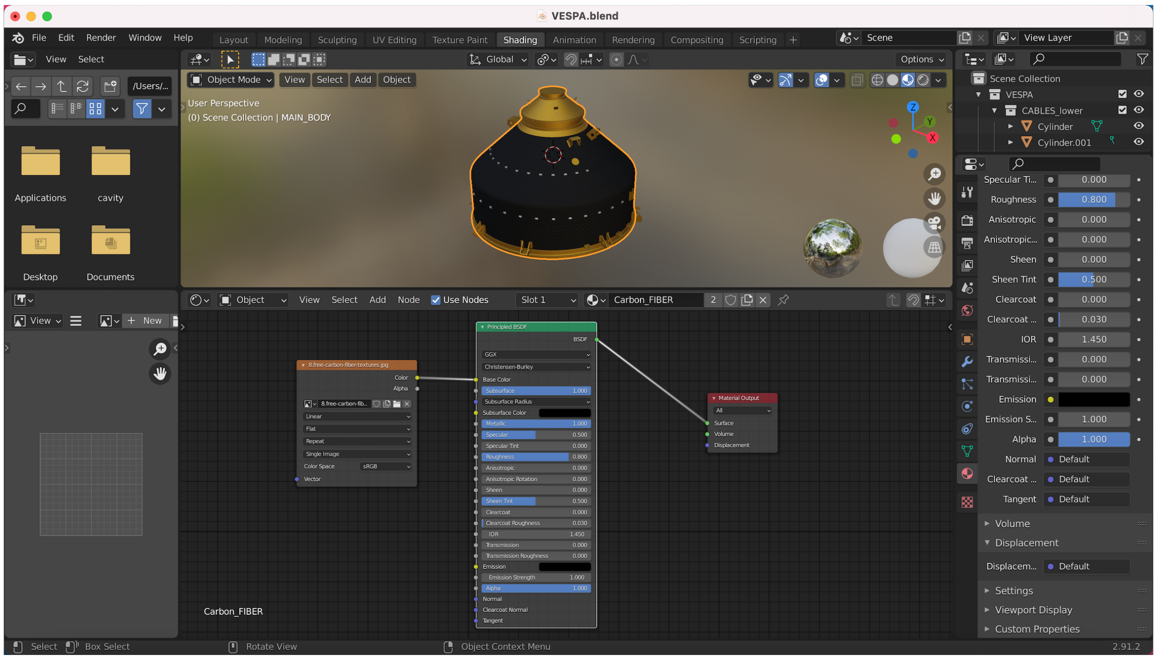


Figure 5.7: *Textures: from the upper-left corner, carbon fiber [61], gold and aluminum*

It can be noticed that the textures for gold and aluminum are not uniform, this could result in a problem for the graphic rendering, especially for the gold one, since the objects at which it is applied are rather big, while for the aluminum, since it is applied to really small objects and the fact that its color gradient is very faint, it will not result in a problem. For the gold texture the problem is solved applying only the lower part of it to the selected objects of the model.

Once the textures are chosen, Blender offer quite a number of parameters to obtain the desired effect of the material, in Fig. 5.8 some of them can be seen. It will be shown only for the Carbon Fiber material since for the others two the concept is the same, only some of the values change.



Specular Ti...	0.000
Roughness	0.800
Anisotropic	0.000
Anisotropic...	0.000
Sheen	0.000
Sheen Tint	0.500
Clearcoat	0.000
Clearcoat ...	0.030
IOR	1.450
Transmissi...	0.000
Transmissi...	0.000
Emission	■
Emission S...	1.000
Alpha	1.000
Normal	Default
Clearcoat ...	Default
Tangent	Default
Displacem...	Default

Figure 5.8: *Node for the creation of the desired material, carbon fiber case*

Then adding some lights, the final result is illustrated in Fig. 5.9. In order to obtain this image, also the camera parameters such as the focal length, the array size etc. should be inserted in blender. The characteristics of the selected camera can be seen in Tab. 5.3

Characteristic	Value
Array size [pixels]	2048x2048
Pixel pitch [μ]	5.5
Frame rate [fps]	5
Focal length [mm]	70
f-number	2.8

Table 5.3: *Camera's characteristics*

From these fundamental ones it is possible to derive other important parameters such as the FOV and the aperture. The pixel pitch is the distance between the centre of each pixel, so, multiplying it by the array size, the sensor size is obtained, $d_{\text{sensor}} = 11.06$ mm, then using the formula seen in Section 3.4, FOV $= 9.034°$. The f-number chosen is 2.8, since as stated in [62] "If you need a fast, low-light lens for astrophotography, then an aperture of f/2.8 or wider is the way to go", and so, following the formula in Section 3.4, $a = 25$ mm.

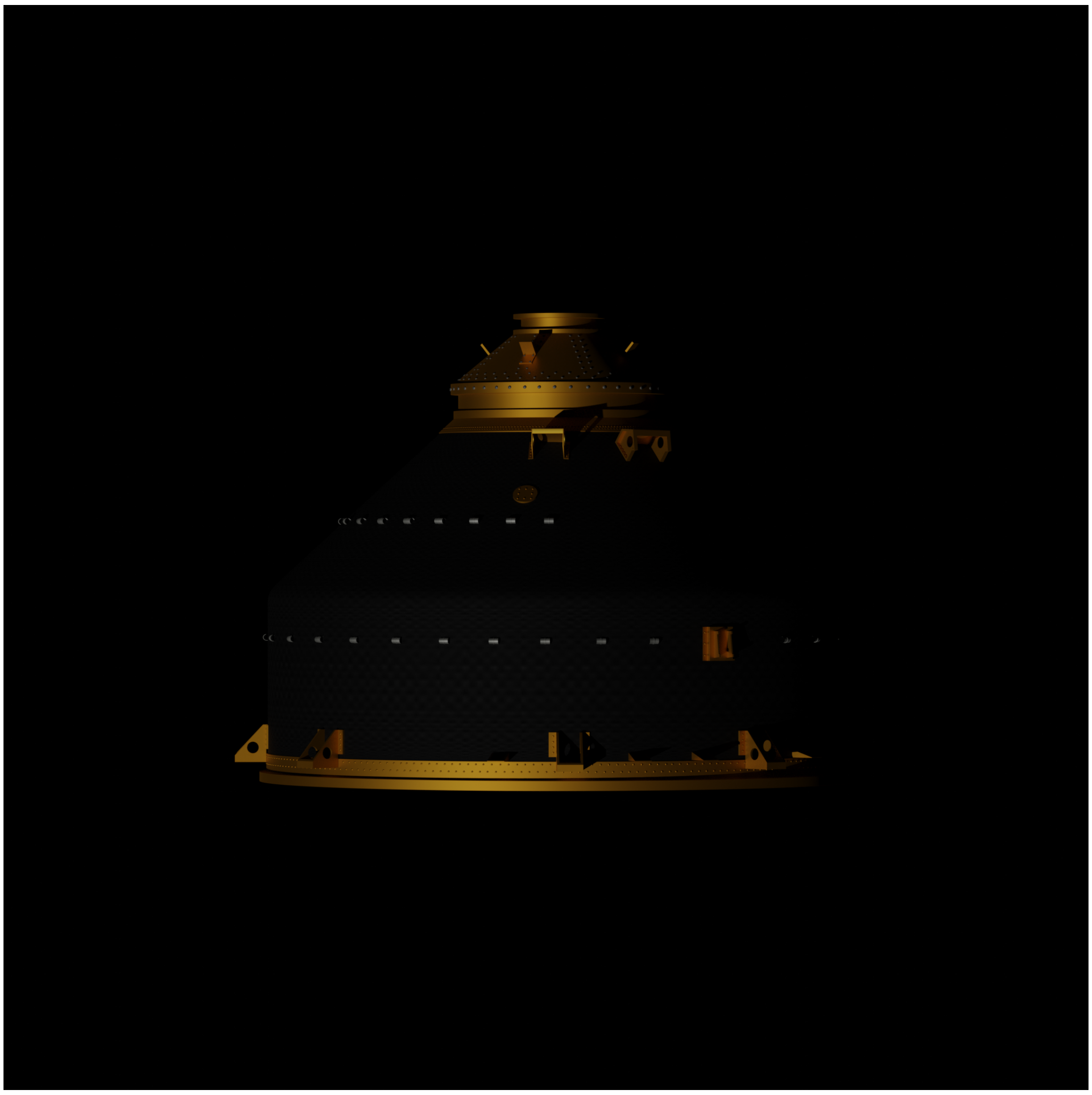


Figure 5.9: *VESPA's complete model*

5.2.2 Environment's model

Once the model of the target has been finished, it's time to think about the environment VESPA is in. The main points are:

- Lighting.
- Earth in the background.

Regarding the lighting, blender is provided of different kind of light, the two most interesting being the *Sun light* and the *Point light*, the first simulates the light coming from the sun as parallel lines and the unite of measure assigned to it is [W/m^2], making the realization of a realistic model possible. The second type considers the light coming from a point source and it radiates outwards in all directions, it is also possible to enlarge this point source making it an emitting light sphere.

Talking about the Earth, some considerations must be made in order to understand the final decision taken. In Fig. 5.10 the chosen model of the Earth, with all its nodes, in the blender environment can be seen.

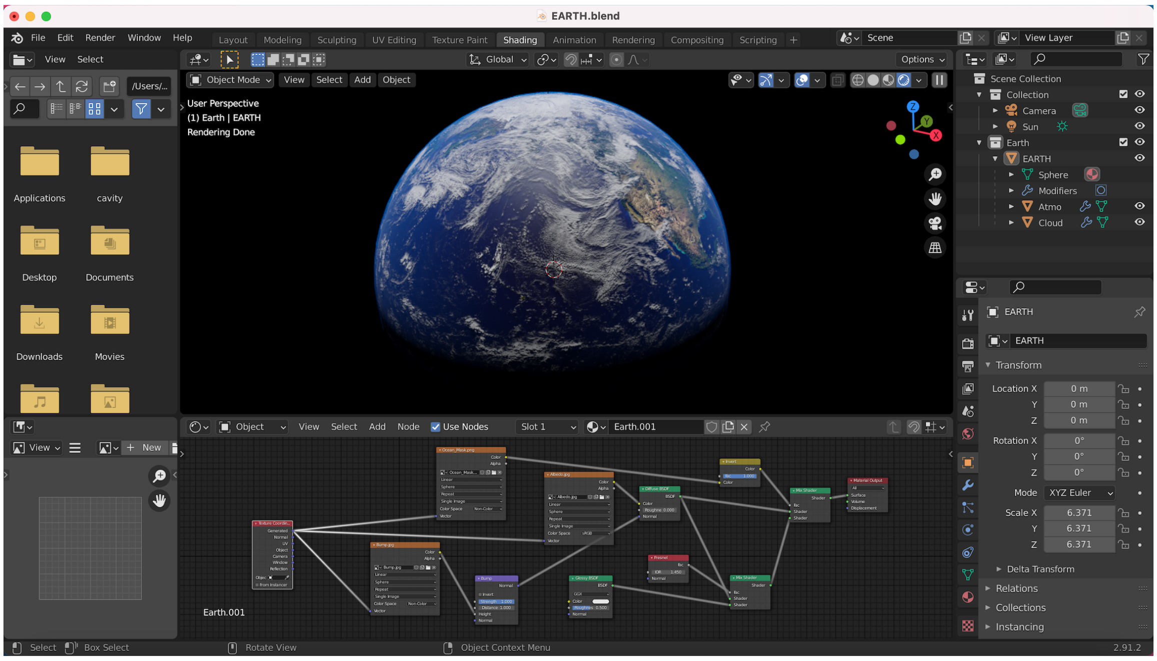


Figure 5.10: *Earth's model in blender*

Looking at the upper-right corner of the image it is possible to notice that the Earth is composed of other two object, *Cloud* and *Atmo*, that, as can be deduced, refer to the clouds and to the atmosphere, their nodes can be seen in Fig. 5.11. All the texture used can be found in [63].

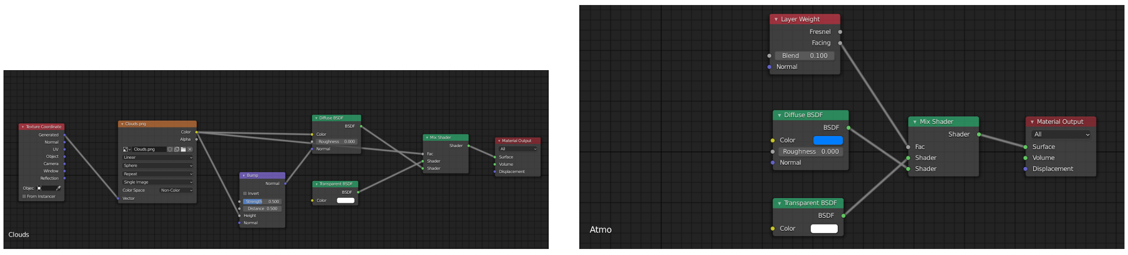


Figure 5.11: *Clouds and Atmosphere model's nodes*

This is one of the most simple and light, in terms of computational power and time needed for rendering, Earth model and although everything seems perfect, once it's time to render, problems arise. The main reason is because of the scale between the target, order of magnitude [m], and the Earth, order of magnitude [10^6 m]. In Blender, for astronomical environment, is possible to set the scale in a such a way that if the scale used is 10000, 1 BU (Blender Unit) = 10000 m, but, even with these changes the problem remains, again, due to the difference in the order of magnitude. In fact, even if a scale of 10^6 is used, although the Earth is fine, to bring the S/C to its real dimension, so meters, the reduction of the size would be too big and its meshes will start to collide and no regular form can be identified. To overcome this problem, a consideration has been made, since the Earth on the background is there only as a disturbing element, there is no need for it to be at its original size, for this reason the scale used is 1000, in such a way the Earth is big enough to cover all the background and the size of the S/C can be its real one, and the same for the relative distances. Summarizing, the scale has been set to 1000, in this way the Earth's radius (in Blender) is 6378 m, the size of the target has been maintained, with a characteristic length of around two meters, the semi-major axis of its orbit is set to 7109 m, in order to have a consistent framework, and the relative distance has been maintained in the order of meters too.

After all these considerations, the model should be complete, but no, the final two problems, that can be seen only after the rendering starts, arise, they are:

- Rendering time.
- Realism of the graphic rendering.

The time needed to render one single image (frame) was more than 30 minutes, the exact time was not verified since during the rendering it is possible to see the image being created and it was not in any way a realistic image, so the process was stopped. The S/C was fine but the Earth in the background was all grainy with the texture being divided in big squares and no feature could be extracted, this is again due to the difference in scale. Due to the enormous rendering time and the horrible results at the end of it, a drastic decision has been taken. For the full trajectory of the camera around VESPA the Earth will not be considered, only its illumination (albedo) will be present. The full work environment is presented in Fig. 5.12.

The light coming from the Sun has been simulated using the *Sun light* present in blender, with an intensity of 1366.9 W/m^2, which is the real intensity of the solar radiation arriving on the Earth, and an inclination of 23.5°. Since the object Earth is not present, no reflection due to it could have been used, for this reason a *Point light* has been used, it has a radius of 6378 m, the one of the Earth, and a power of 280 MW, in this way the power radiated outwards is around 40 % of the one coming from the sun, simulating the albedo effect, and a faint azure color has been used. Also the Sun, as an object, has been placed in the work environment, proportionate accordingly to the distance at which it is set. This has been done since, as can be seen also on Earth, if the Sun enters the FOV of the camera it will saturate the image and nothing would be identified. For the chosen trajectory this does not happen and it's not certainly an isolated case, since, with just some calculation it can be verified if the Sun enters the camera's FOV or not.

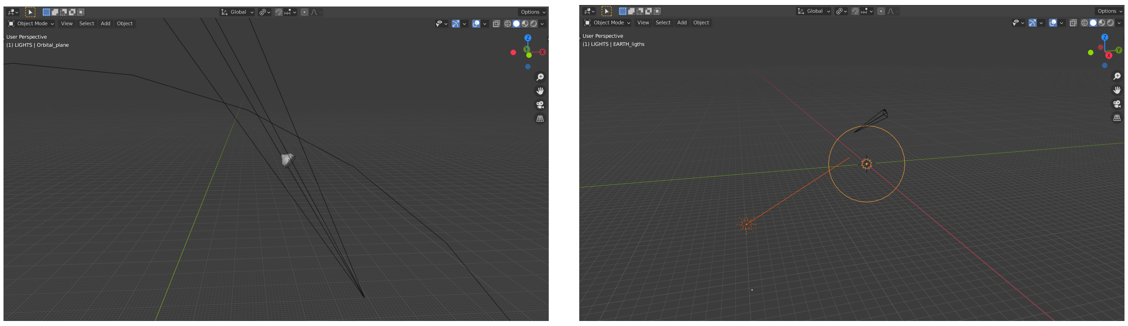


Figure 5.12: *Full model*

To end this discussion, just two words will be spent on the trajectory chosen. The point in which VESPA is located along its orbit is identified by a true anomaly, θ, of 80°, and the chosen trajectory in an helical one that develops along the direction of the velocity. The camera starts from a distance of around 30 m from the the S/C.

5.3 Camera calibration step

For the camera calibration the camera calibration app [64] present in matlab [65]. In order to use the following workflow should be respected:

1. Prepare images, camera, and calibration pattern.
2. Add images and select standard or fisheye camera model.
3. Calibrate the camera.
4. Evaluate calibration accuracy.
5. Adjust parameters to improve accuracy (if necessary).
6. Export the parameters object.

To better the results, a number of images of the calibration pattern between 10 and 20 should be used. The calibrator requires at least three images. The calibration pattern and the camera setup must satisfy a set of requirements to work with the calibrator. The calibration pattern most used is the checkerboard, a sequence of black and white squares, it can be seen in Fig. 5.13.

They have to satisfy the following requirements:

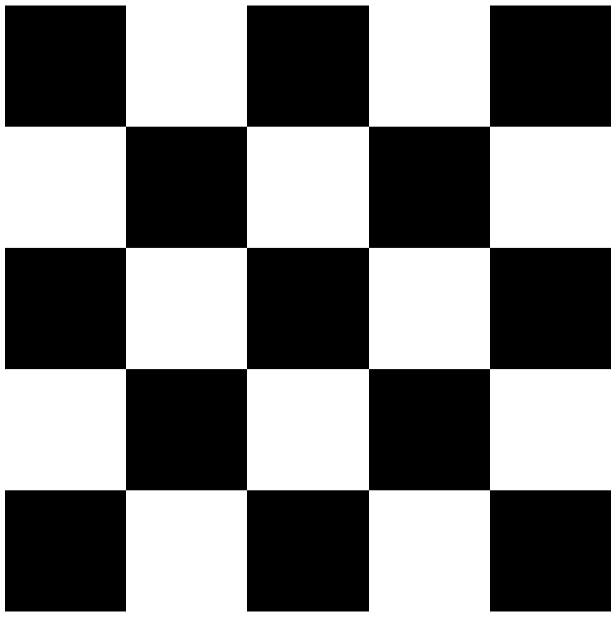


Figure 5.13: *Checkerboard pattern*

- Containing an even number of squares along one edge and an odd number of squares along the other edge.
- They have to be non-square pattern, and must contain two black corners along one side and two white corners on the opposite side.

This criteria enables the app to determine the orientation of the pattern and the origin. The calibrator assigns the longer side to be the x-direction. A square pattern can produce unexpected results for camera extrinsics. It can be noticed that the checkerboard in Fig. 5.13 doesn't respect any of the above requirements and so, obviously, won't be used for the calibration.

Another important parameter that must be introduced is the size of one square, the checkerboard used has a square of 35 mm, and it is shown in Fig. 5.14.

The images of the pattern captured should be at a distance roughly equal to the distance from the camera to the objects of interest. For example, if the plan is to measure objects from 2 meters, the pattern must be kept approximately at 2 meters from the camera. In addition the checkerboard should be placed at an angle less than 45 degrees relative to the camera plane. It is important to capture a variety of images of the pattern. Lens distortion increases radially from the center of the image and sometimes is not uniform across the image frame. To capture this lens distortion, the pattern must appear close to the edges of the captured images. As a general rule, the checkerboard should fill at least 20% of the captured image.

To begin calibration, images must be added. It is possible to add saved images from a folder or add images directly from a camera. The calibrator analyzes the images to

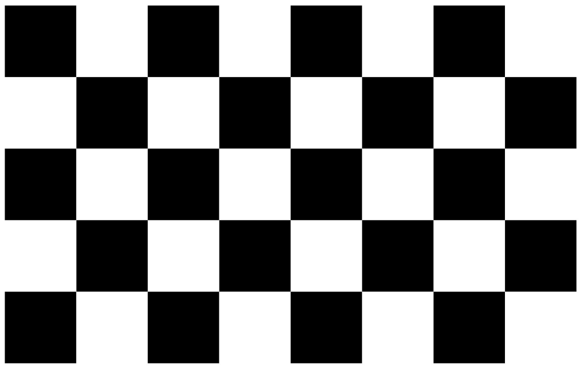


Figure 5.14: *Checkerboard used*

ensure they meet the calibrator requirements. The calibrator then detects the points on the checkerboard. After the images are added, the checkerboard Square Size dialog box appears. The size of the checkerboard square must be specified by entering the length of one side of a square from the checkerboard pattern. Once the accepted images are satisfactory, the real calibration can start. The default calibration settings assume the minimum set of camera parameters. Start by running the calibration with the default settings. After evaluating the results, it can be tried to improve calibration accuracy by adjusting the settings and adding or removing images and then calibrating again.

It is possible to evaluate the calibration accuracy by examining the reprojection errors, examining the camera extrinsics, or viewing the undistorted image. To improve the calibration, high-error images can be removed, more images can be added, or the calibrator settings could be modified.

5.4 V-SLAM algorithm

. In this section the V-SLAM algorithm utilized will be analyzed and discussed. As already said through the introduction and in general the whole thesis, the algorithm utilized is a features-based one and in particular the method used to detect and extract features is the ORB one, making it an ORB-SLAM algorithm.

In Fig. 5.15 an overview of the whole process is present, it is formed by four main steps:

- **Map initialization**: ORB-SLAM starts by initializing the map of 3-D points from two video frames. The 3-D points and relative camera pose are computed using triangulation based on 2-D ORB feature correspondences.

- **Tracking**: Once a map is initialized, for each new frame, the camera pose is estimated by matching features in the current frame to features in the last key frame. The estimated camera pose is refined by tracking the local map.

- **Local mapping**: The current frame is used to create new 3-D map points if it is identified as a key frame. At this stage, bundle adjustment is used to minimize reprojection errors by adjusting the camera pose and 3-D points.

- **Loop closure**: Loops are detected for each key frame by comparing it against all previous key frames using the bag-of-features approach. Once a loop closure is detected, the pose graph is optimized to refine the camera poses of all the key frames.

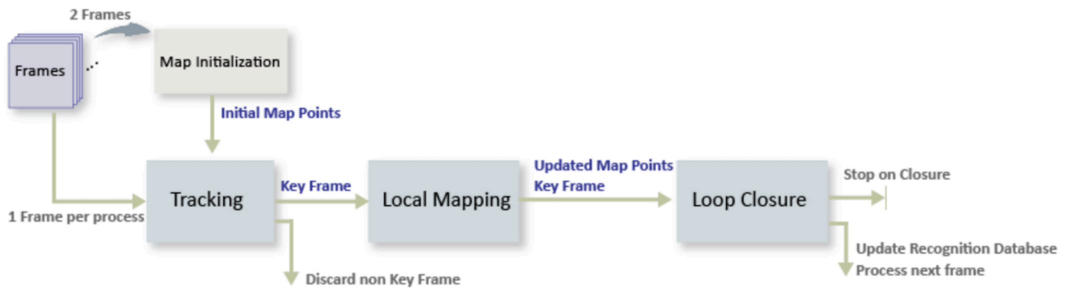


Figure 5.15: *ORB-SLAM overview*

5.4.1 Map initialization

The ORB-SLAM pipeline starts by initializing the map that holds 3-D world points. This step is crucial and has a significant impact on the accuracy of final SLAM result. Initial ORB feature point correspondences are found by matching them between a pair of images. After the correspondences are found, two geometric transformation models are used to establish map initialization:

- **Homography**: If the scene is planar, a homography projective transformation is a better choice to describe feature point correspondences.

- **Fundamental matrix**: If the scene is non-planar, a fundamental matrix must be used instead.

The model that results in a smaller reprojection error is selected to estimate the relative rotation and translation between the two frames. Since the RGB images are taken by a monocular camera which does not provide the depth information, the relative translation can only be recovered up to a specific scale factor. Given the relative camera pose and the matched feature points in the two images, the 3-D locations of the matched points are determined using a triangulation function. A triangulated map point is valid when it is located in the front of both cameras, when its reprojection error is low, and when the parallax of the two views of the point is sufficiently large. Then the two keyframes and the corresponding map points are stored.

5.4.2 Tracking

The tracking process is performed using every frame and determines when to insert a new key frame. Each frame is processed as follows:

1. ORB features are extracted for each new frame and then matched with features in the last key frame that have known corresponding 3-D map points.

2. Estimate the camera pose with the Perspective-n-Point algorithm.

3. Given the camera pose, project the map points observed by the last key frame into the current frame and search for feature correspondences.

4. With 3-D to 2-D correspondence in the current frame, refine the camera pose by performing a motion-only bundle adjustment.

5. Project the local map points into the current frame to search for more feature correspondences and refine the camera pose again using the motion-only bundle adjustment.

6. The last step of tracking is to decide if the current frame is a new key frame. If the current frame is a key frame, continue to the **Local Mapping** process. Otherwise, start **Tracking** for the next frame.

In order to decide if the current frame is a keyframe or not , two conditions must be satisfied:

- At least a defined number of frames have passed since the last key frame or the current frame tracks fewer than 100 map points.

- The map points tracked by the current frame are fewer than 90% of points tracked by the reference key frame.

5.4.3 Local mapping

Local mapping is performed for every keyframe. When a new one is determined, add it to the keyframes and update the attributes of the map points observed by it. To ensure that the map points contains as few outliers as possible, a valid map point must be observed in at least 3 keyframes. New map points are created by triangulating ORB feature points in the current keyframe and its connected keyframes. For each unmatched feature point in the current one, search for a match with other unmatched points in the connected keyframes. The local bundle adjustment refines the pose of the current keyframe, the poses of connected keyframes, and all the map points observed in these key frames. This is the step that requires the highest time to be executed, especially the local bundle adjustment algorithm, it can be seen in Section 6.

5.4.4 Loop closure

The loop closure step takes the current keyframe processed by the local mapping process and tries to detect and close the loop. Loop detection is performed using the bags-of-words approach. A visual vocabulary represented as a bagOfFeatures object is created offline with the SURF descriptors extracted from a large set of images in the dataset by the SURF feature extractor function. The loop closure process incrementally builds a database that stores the visual word-to-image mapping based on the bag of SURF features. Loop candidates are identified by querying images in the database that are visually similar to the current keyframe. A candidate keyframe is valid if it is not connected to the last one and three of its neighbor keyframes are loop candidates. When a valid loop candidate is found, the relative pose between the loop candidate frame and the current keyframe is computed. The relative pose represents a 3-D similarity transformation. Then add the loop connection with the relative pose and update the map points and keyframes set.

It must be noted that in the case study of this thesis no points in space are visited twice, due to the characteristic of the chosen trajectory, and so a loop closure will never happen.

A last passage, not present in the general overview, and not possible during real time operations since it is performed after all the analysis is complete and all the frames have been used, is the similarity pose graph optimization. It is performed over the essential graph in order to correct the drift of camera poses. The essential graph is created internally by removing connections with fewer than a minimum number of matches in the covisibility graph. After the similarity pose graph optimization, the 3-D locations of the map points are updated.

6 Results

All the results obtained are here reported. It starts with results from the camera calibration, following its various step, refining the various parameters. Then the V-SLAM results are presented, including the various time for the execution of the single parts of the algorithm and the error between the reconstructed trajectory and real one.

6.1 Camera calibration results

The checkerboard used has already been introduced in Section 5.3. A part of the set of images used for the calibration is shown in Fig. 6.1.

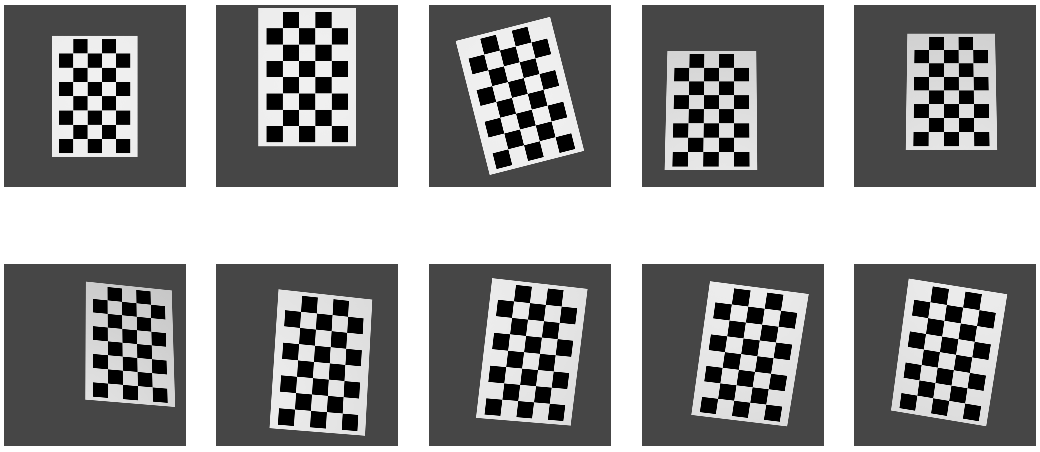


Figure 6.1: *Part of the set of image of the checkerboard used for calibration*

6.1.1 First calibration

As stated in Section 5.3 in the first calibration all the images will be used, no initial condition for the K matrix will be inserted and since it is not the case of a fisheye lens, only two radial distortion parameters will be calculated In Fig. 6.2 the first two images of the set are shown after the tool has detected the grids point, its origin and the axes.

while the reprojected points, again for the first two images of the set, can be seen in Fig. 6.3.

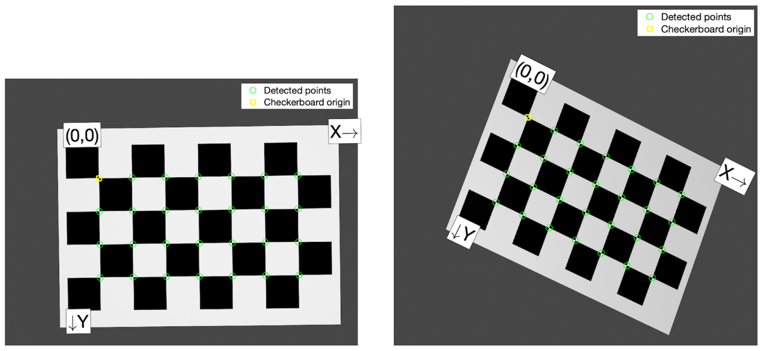


Figure 6.2: *Detected grid points for the first two images of the set*

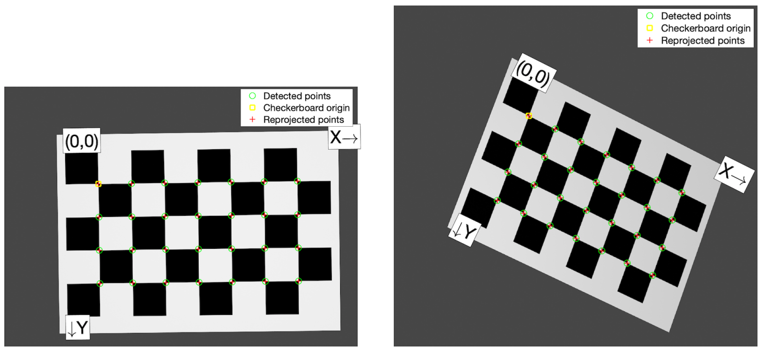


Figure 6.3: *Reprojected grid points for the first two images of the set*

In Fig. 6.4 the mean reprojection error per image is shown, while in Fig. 6.5 the position of the pattern, keeping the camera fixed, on the left, and the position of the camera keeping the pattern fixed, on the right, are shown.

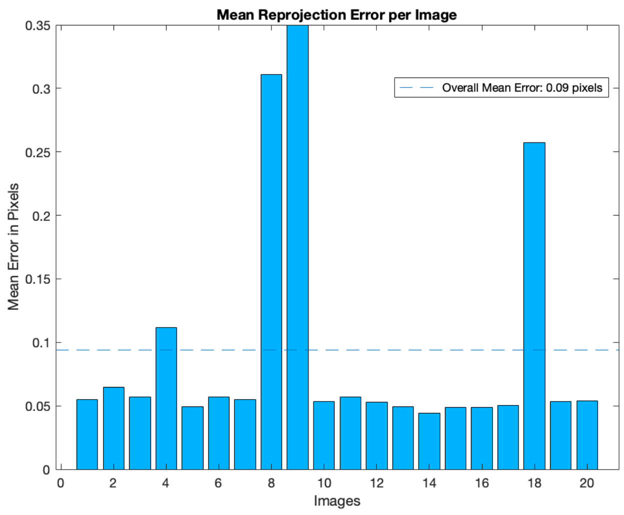


Figure 6.4: *Mean reprojection error per image on the first run*

The radial distortion calculated are $k_1 = 0.0450$ and $k_2 = -10.3502$, instead, for what regard $\mathbf{K}$, the camera intrinsic matrix, it is:

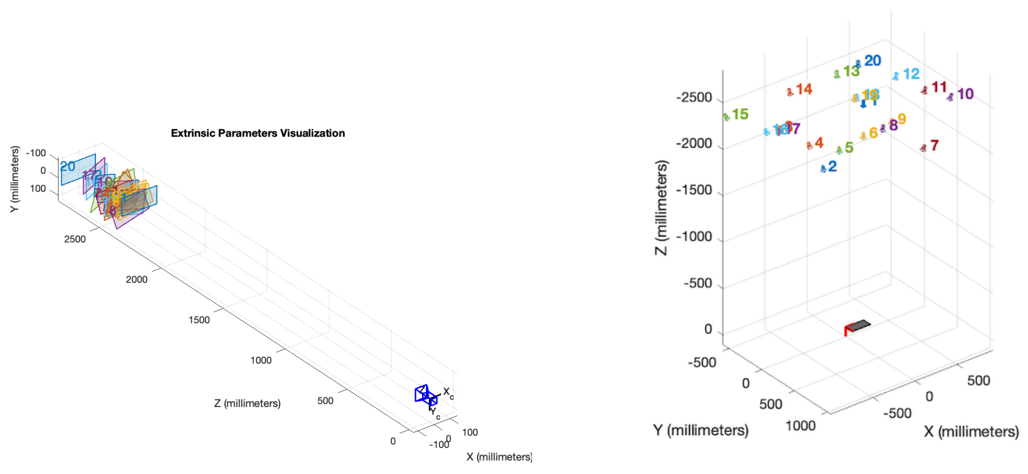


Figure 6.5: *Extrinsic parameters visualization*

$$\mathbf{K} = \begin{bmatrix} 12980.82 & 0 & 1027.45 \\ 0 & 12980.82 & 1030.22 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.1}$$

6.1.2 Second calibration

Here the three images that gave the highest mean reprojection error are eliminated, bringing the Overall Mean Error to 0.06 pixels. The obtained radial distortion coefficients are $k_1 = 0.0484$ and $k_2 = -9.84$ and $\mathbf{K}$:

$$\mathbf{K} = \begin{bmatrix} 12959.1 & 0 & 1015.6 \\ 0 & 12959.1 & 1039.5 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.2}$$

6.1.3 Third calibration

Now an initial guess is introduced, since the focal length and the array size are available, the ideal $\mathbf{K}$ should be the following:

$$\mathbf{K}_{\text{ideal}} = \begin{bmatrix} 12727.27273 & 0 & 1024 \\ 0 & 12727.27273 & 1024 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.3}$$

Then putting it inside the camera calibration app of Matlab, the radial distortion coefficients are $k_1 = 0.0484$ and $k_2 = -9.839$, and $\mathbf{K}$ is:

$$\mathbf{K} = \begin{bmatrix} 12958.6 & 0 & 1015.5 \\ 0 & 12958.6 & 1039.5 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.4}$$

6.1.4 Fourth calibration

Here, the final calibration is presented. The last image with the highest Mean reprojective error is eliminated. As this is the last case, in Fig. 6.6, the mean reprojection error per image will be shown again.

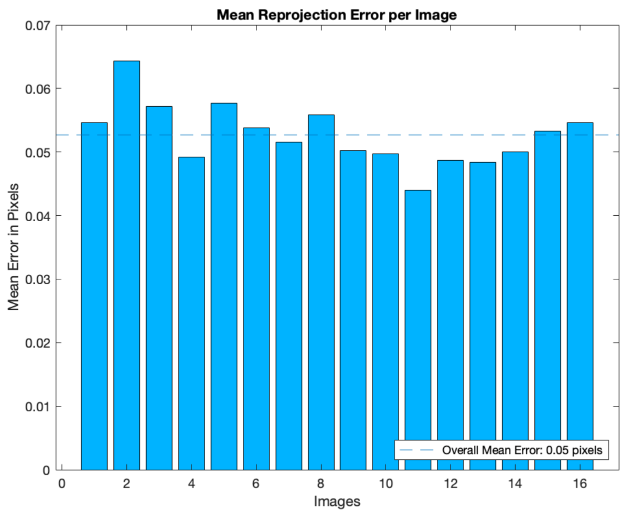


Figure 6.6: *Mean reprojection error per image on the fourth run*

The radial coefficients obtained are $k_1 = 0.0511$ and $k_2 = -10.3651$ and the intrinsic camera matrix $\mathbf{K}$ is:

$$\mathbf{K} = \begin{bmatrix} 12948.86 & 0 & 1017.8 \\ 0 & 12948.86 & 1035.9 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.5}$$

It must be said that in the case of synthetic images the radial distortion should not be present as the images are generated through an ideal model of camera and the K matrix should have been reconstructed perfectly as the ideal one, shown in Eq. 6.3. Evidently this is not the case, but not everything is lost, this can be interpreted in two ways. The first, and most immediate one, is that the camera calibration app made a mistake, but even if this is the case, since in reality there are always some errors, this could be applied in order to take them into account. The second one is that Blender made some mistakes in the rendering of the images and so, in reality, the camera calibration app reconstructed $\mathbf{K}$

in a good way, since the images were at fault. Regardless of which tool made the mistake, it has been decided to use the **K** matrix of the fourth calibration, the closest to the ideal one, in the ORB-SLAM algorithm to make the case study as real as possible.

6.1.5 Extra: tangential distortion coefficients

As an extra, also the case considering the tangential distortion coefficients is reported but this will not be used for the future discussion.

The radial distortion coefficients are $k_1 = 0.0575$ and $k_2 = -12.2534$ and the tangential one are $p_1 = -2.684 * 10^{-4}$ and $p_2 = 4.96 * 10^{-4}$. While the first set is similar to other case, a bit bigger, the values of the tangential distortion coefficients are very low and can be easily neglected. Regarding **K**, it is:

$$\mathbf{K} = \begin{bmatrix} 12949.32 & 0 & 1034.5 \\ 0 & 12949.32 & 1028.2 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.6}$$

6.2 Algorithm validation

Before starting the feature extraction and the reconstruction of the camera's trajectory around the target, VESPA, a test, in order to verify the effectiveness of the tools used and the correctness of the results achieved, are performed. It has been decided to carry out this validation tests through the "RGB-D dataset and benchmark" of the computer vision group, TUM, department of informatics, Technical University of Munich [66]. The sequence of images used is called *long office household*, the sensor/camera was moved along a large round through a household and office scene with much texture and structure.

All the camera parameters are given and **K** is:

$$\mathbf{K} = \begin{bmatrix} 535.4 & 0 & 320.1 \\ 0 & 539.2 & 247.6 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.7}$$

The results of this validation test are presented and, as can be seen from Fig. 6.7 the reconstructed trajectory is very close to the real one and the error has an RMSE of 0.13 m. So, it can be concluded that, in this first test, the algorithm is effective and the test is successfully carried out.

Although successful, it must be pointed out that this is just a first validation test and even if it guarantees, to a certain extent, that the algorithm works fine, it doesn't take into account all the possible variations of the algorithm's settings and of the scenarios. In order to fully validate it, an extended campaign of tests is necessary, taking into account the previously mentioned variations.

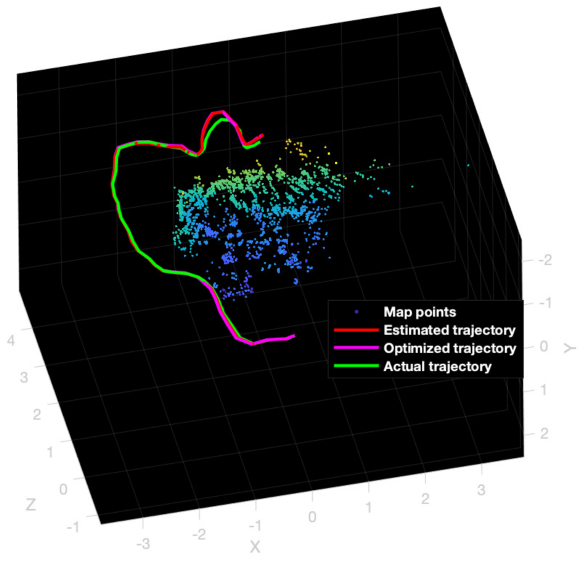


Figure 6.7: *Comparison between the reconstructed trajectory and the real one in the validation test.*

6.3 V-SLAM results

In the following subsections test cases of the V-SLAM algorithm are reported. It must be remembered that the CPU of the computer utilized, a 2,2 GHz Intel Core i7 quad-core, has been capped at 40% of its full capability, this is important especially for the execution times calculated. Moreover, the following cases are only a small portion of the possible ones, and even here, in order to obtain more consistent and truthfully results, a sensitivity analyses of all the parameters of the algorithm, of the initial conditions and of the scenario, should be performed. In fact, although some good results are obtained and some considerations are made, they refer only to the cases studied, and should be verified for a general view.

Before exploring the results obtained, more information, with respect to Section 5.2, about the selected trajectory must be said.

6.3.1 Trajectory

As said before, the chosen shape for the trajectory is an helical one, to be more precise, the camera will do two turns around VESPA, so a full rotation of 720°. The direction in which the helical develops is that of the velocity of the S/C and the starting point of the trajectory is at a distance of 30 m behind VESPA, it can be seen in Fig. 6.8.

In Blender it has been defined as an animation of 721 frames, so a frame for each degree of the rotation, since the first frame is at 0°

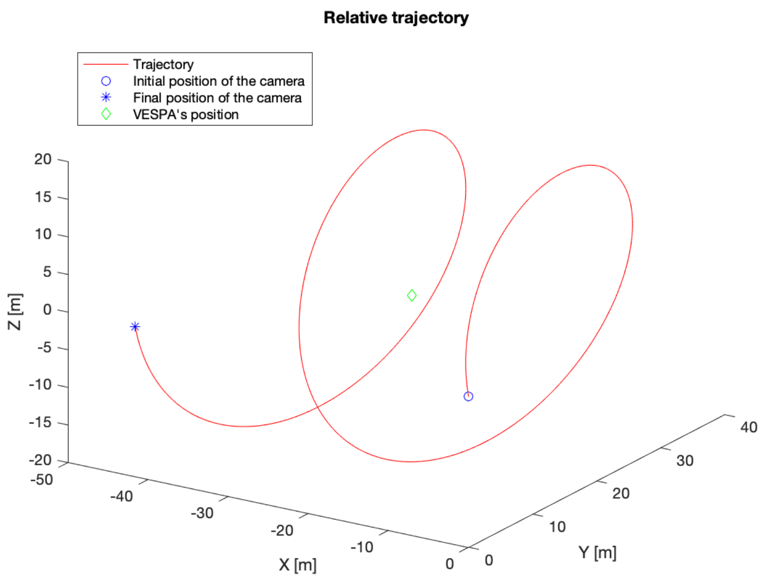


Figure 6.8: *Trajectory of the camera around VESPA*

6.3.2 Test case 1

The test cases are divided by the orientation of VESPA along its orbit. Since data about its attitude have not been found, it has been decided to study two cases, one in which the orientation of VESPA's z axis is along the velocity direction and one in which it is perpendicular. In this subsection the first case is studied, but two sub-cases are presented, one with the full trajectory and one with only half of the trajectory. The reasons that brought this division will be later on explained

6.3.2.1 Full Trajectory

Here, differently from the following subsection, a digression about the map initialization step and the angle view between two images is present, this is due to the fact that the ORB-SLAM algorithm is not able to reconstruct in a good way the trajectory, or at least, part of it. In Fig. 6.9 the first frame is shown while in Fig. 6.10 the map initialization is reported, it happens with the first two frames of the animation.

Then the entire algorithm is put into work and the reconstructed trajectory is obtained, in Fig 6.11 it, red line, can be seen plotted against the real one, green line, and it can be immediately noticed that something is not right. The biggest problem is at the beginning of the reconstruction. Even though the camera is not exactly behind the S/C, the algorithm thinks that it is and start its mapping from there, then after some keyframes are examined it tries to come back, as near as possible, to the real one, but the error made is already too big.

Since the reconstruction fails, the time necessary to the algorithm to map each keyframe will not be discussed.

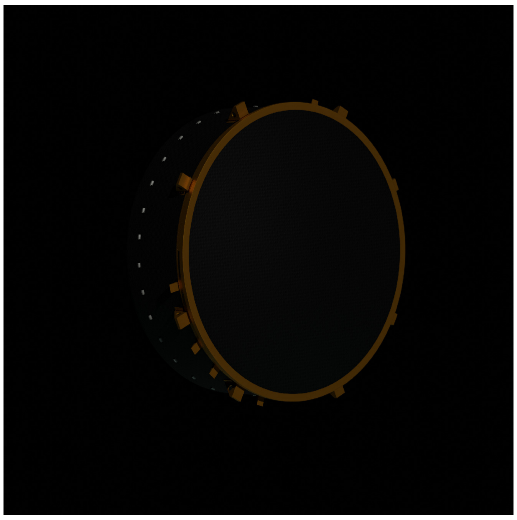


Figure 6.9: *First frame*

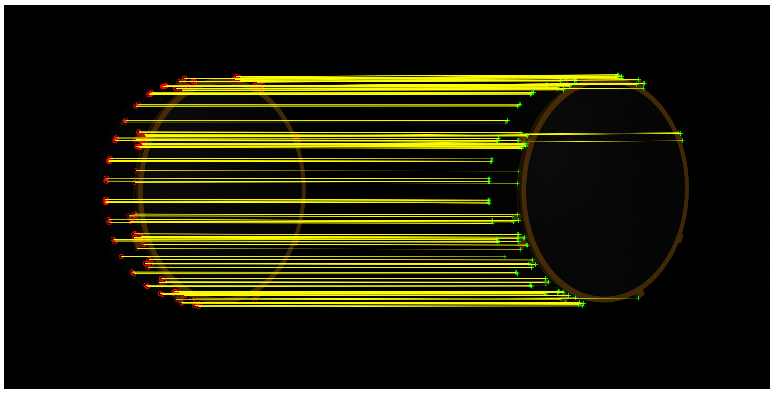


Figure 6.10: *Map initialization matching*

As said in Section 5.4.1, the map initialization step is fundamental, since it is the step in which the map is generated for the first time. Since it will be used for navigation in the following steps, it is important to achieve an estimation of its 3D points as accurate as possible. The separation angle between the two frames used for initialization is the key

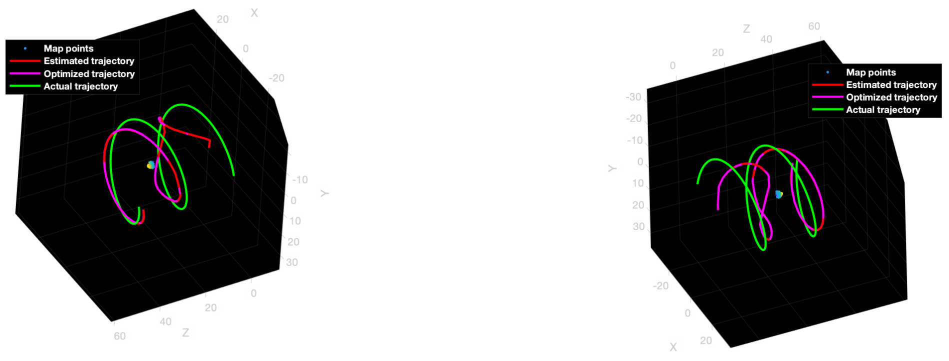


Figure 6.11: *Comparison between the reconstructed trajectory and the real one*

by which this problem could be unlocked. In Fig. 6.12 the effect of the difference in the viewing angle between the images can be appreciated.

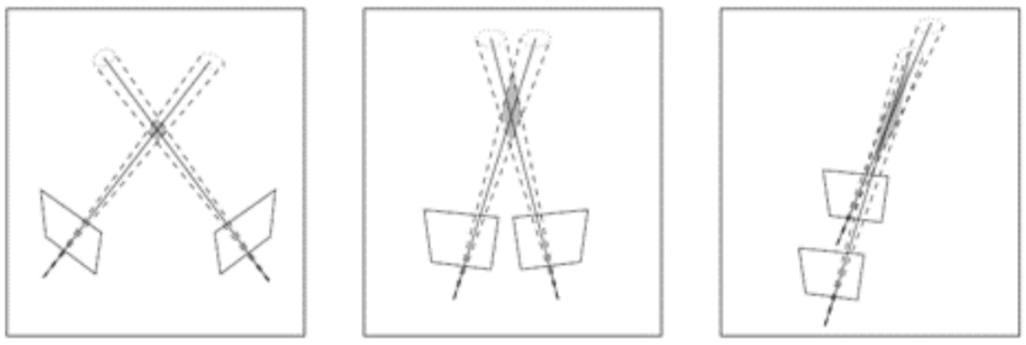


Figure 6.12: *Uncertainty on 3D points due to the angle between the views*

Very low difference in the separation angle will result in the failure of the method since the uncertainty of the 3D points location would be too high. Moreover, also moving in a straight trajectory, with respect to the target, doesn't allow any perspective view of it and so the map will not be be correctly generated. In fact, particular steps should be followed, first an investigation of the object, then a study on how the approach phase would be performed and so understand when to let the map initialization step begin.

It could be thought that the higher the viewing angle the better the map initialization step, this could be right for an human eye but not for a computer and the computer vision. This is due to the fact that the method utilized is a feature-based one and in order to reconstruct the 3D points features must be matched/tracked. It is generally assumed that features don't change their appearance by large amounts, obviously not for the whole trajectory/video but around a selected frame, and so they can be matched accurately and the tracking could be performed. But if a large view angle is used and so the frames examined are very different, the features matched will be inevitably low and the accuracy of the reconstruction of the camera pose will be low too. In general, a trade-off between the previously cited effect should be sought.

In order to understand if the problem for the map initialization previously mentioned is the fact that the camera is behind VESPA and other than a rotation motion, also a translation is present, making the case similar to the one where the camera moves in straight trajectory towards the target, or if it simply a too low angle view between the frames, a study has been carried out. The part of the trajectory considered is only the one of the first 100 frames, since the biggest problem is a the beginning and that is the point of interest.

The condition for the map initialization step is that a minimum number of features, 100, must be matched, keeping this condition, another one has been added, the number of the frame that must be used. Since in the previous case the frames used for the map initialization are the first two, this number will start from three and will be increased until the condition of the minimum number of matches is no longer satisfied, obviously this can be changed but, in order to have a good reconstruction of the camera pose it won't be done, or if by increasing it again the quality of the reconstruction drops.

The first test is performed by imposing that the initialization can be performed by starting from the third frame. The initialization is successful with the third frame but as can be seen from Fig. 6.13 the reconstruction of the trajectory faces the same problem as before.

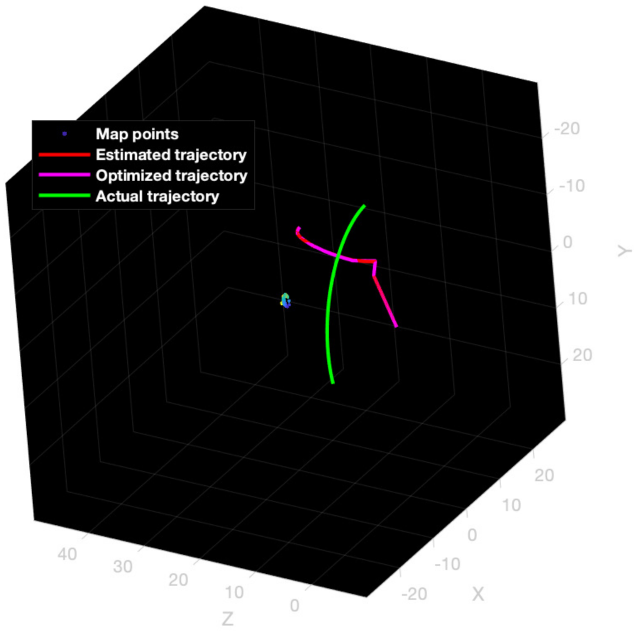


Figure 6.13: *Comparison between the reconstructed trajectory and the real one with map initialization performed by the first and third frames*

Instead, by imposing that at least three frames must pass, so starting from the fourth one, the initialization doesn't happen until the eleventh frame, this is due to the fact that not enough matches are found until that frame and the results change completely. In Figs. 6.14, 6.15 the initialization step matching and the reconstructed trajectory are shown.

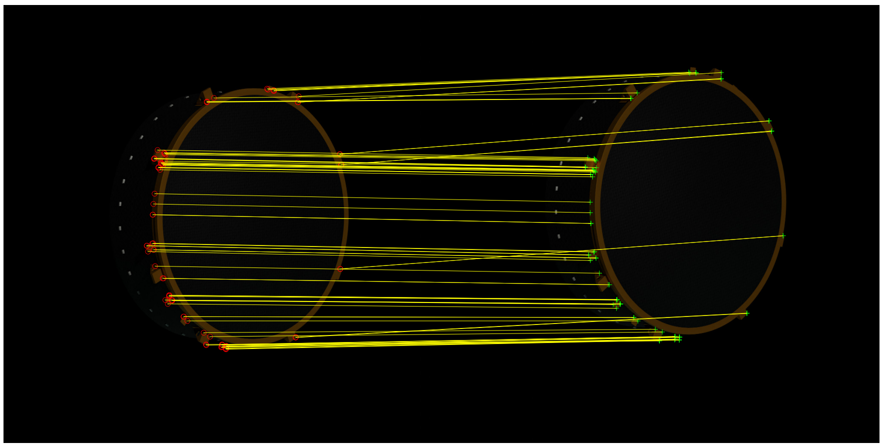


Figure 6.14: *Map initialization step between the first and the eleventh frame*

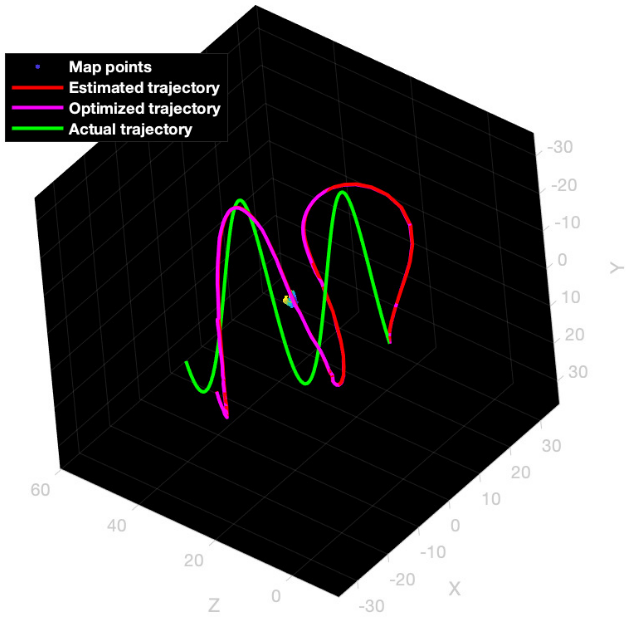


Figure 6.15: *Comparison between the reconstructed trajectory and the real one with with map initialization performed by the first and the eleventh frames*

As can be seen from them, with this map initialization, the trajectory is reconstructed in a better way than before, at least the general shape is present. But, the reconstruction error is very high, having a RMSE of 6.162 m. Just from this result it could be concluded that the problem with the initialization was in fact the starting position of the camera, behind the target, with its motion, but, more importantly, the angle view, that at the beginning was too small. In fact, increasing again the number of the frame for the map initialization, but keeping the minimum number of matches at 100, the next useful frame is the 16th one. With this initialization, the reconstruction becomes even better, having an RMSE of 1.6221 m, and can be appreciated in Fig. 6.16.

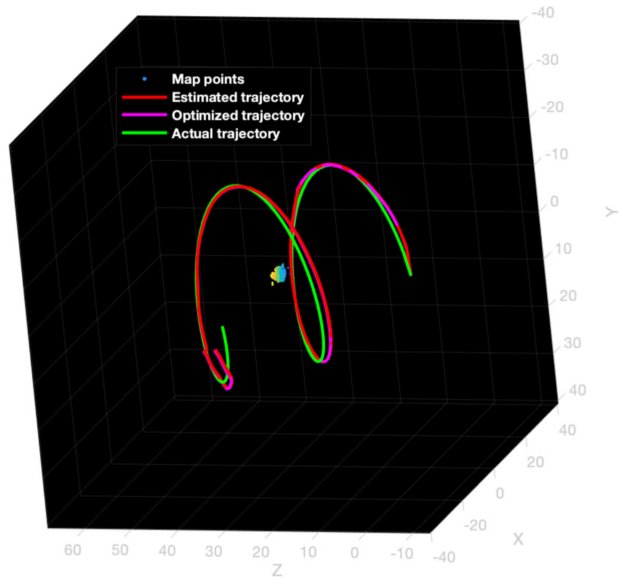


Figure 6.16: *Comparison between the reconstructed trajectory and the real one with with map initialization performed by the first and the 16th frames*

Increasing again the angle view, the map initialization is performed with the 23th frame but the reconstruction of the initial step is not good validating the previous statement than an angle view too large is not good.

As already explained in Section 5.4, there are two conditions that must be satisfied in order for the algorithm to recognize a frame as a keyframe, one is based on number of matched feature and one on the number of frames that has passed from the last keyframe. During the simulation it has been noticed that setting this last criteria as small as possible won't always give better results, and in reality it is the total opposite, making also the algorithm stop in certain cases.

For this reason an analysis between the results obtained, time needed and reconstruction error, between the various frame-step, number of frames that must pass from the last keyframe, is performed. The results obtained point out that starting from a frame-step of 1 and increasing it, the results keep becoming better, arriving at a maximum for the frame-step of 7 and 8, after these values the reconstruction error and the time needed keep getting worse. Frame-step 7 and frame-step 8 maximize the performance of two different parameters, for the latter the times needed to the algorithm to work are minimized, while for the former the reconstruction error is minimized. These results can be brought back again, to a certain extent, to the angle views between two images, if too small, the algorithm is not able to properly reconstruct the 3D points and the camera pose, and also to particular sections of the algorithm that will be later explained.

Since by using a frame-step of one the algorithm stops prematurely, it will be shown in the following subsection, the first frame-step analyzed is two. The results obtained are shown in Figs. 6.17, 6.18, 6.19.

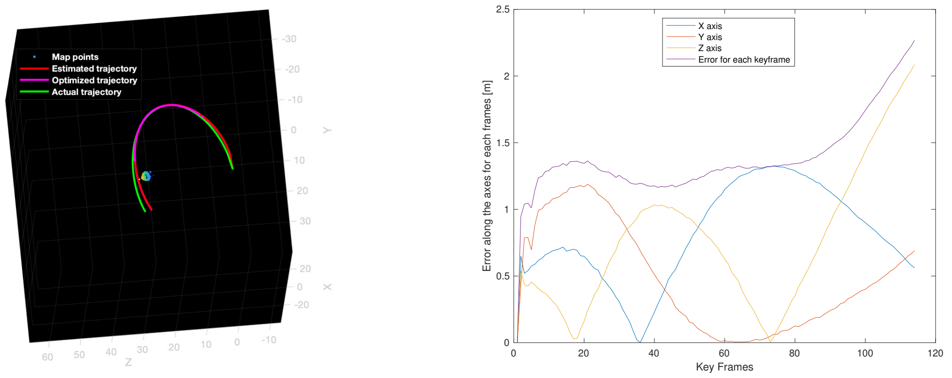


Figure 6.17: *2 FS: Comparison between the reconstructed and real trajectory (left), Error (right)*

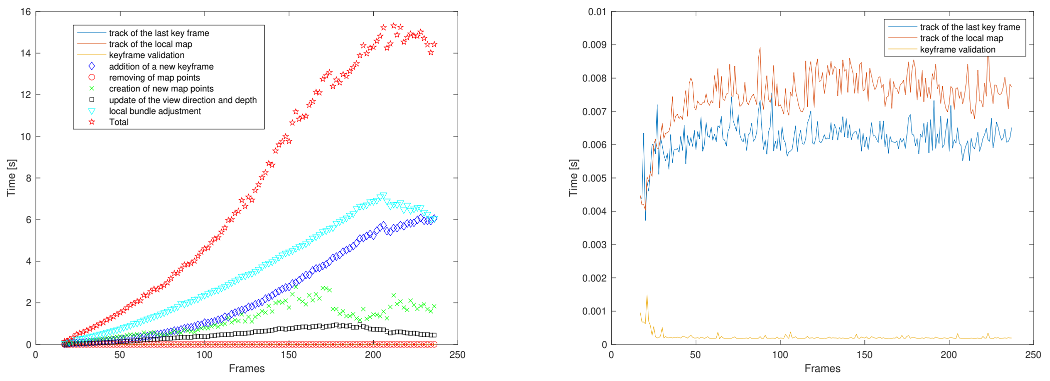


Figure 6.18: *2 FS: Important times during the tracking (left), closeup to the first three important times that are not visible (right)*

Before discussing those results, the meaning of these graphs should be explained. Starting from Fig. 6.17, on the left, the reconstructed trajectory is plotted against the real one, also the optimized one is present but it is not interesting in this discussion, on the right side instead, the error, along the three axis and its normal, between the two trajectories is presented. In Fig. 6.18 the execution times are reported, on the left side, the times needed by each part of the algorithm, in order, are illustrated. Starting from

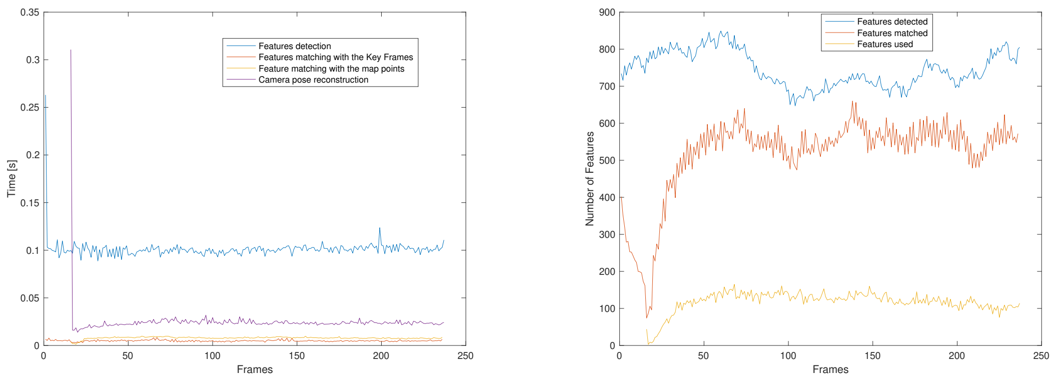


Figure 6.19: *2 FS: Fundamental times (left), Number of features (right)*

the time necessary for tracking the last keyframe, then the local map, so the keyframe validation and if valid its addition to the keyframe set, after this, the time needed for the elimination of map points that don't appear in more than 3 keyframes, subsequently the one for the creation of new map points that were not yet created, then the time for the update of the view direction and depth of the camera and lastly for the local bundle adjustment. Then the sum of all this times is presented. On the right side, since the scale of the first three times listed is way smaller than the one of the others and are not clearly visible, a close up of them is shown. Going to the last one, Fig, 6.19, on the left, the time needed for the process of feature detection, feature matching and camera pose reconstruction are shown, while on the right, the number of features detected, matched, and the ones used for camera pose reconstruction are illustrated. The number of features that can be detected has been capped at 1000.

As can be immediately seen from Fig. 6.17, the trajectory has not been fully reconstructed, but not because there was an error in the algorithm and so it stopped naturally, but it was forcefully stopped by the author because the time needed for the execution at each step was becoming higher and higher, as can be verified by Fig. 6.17, on the left. Keeping the eyes on the same figure, the main culprits can be immediately identified. As expected, since it is stated in literature, the process that takes the most execution time is the bundle adjustment, even if it is in its local version, then the next one, that kept increasing and was becoming even more important than the local bundle adjustment is the addition of a new keyframe in the keyframe set. This is due to the fact that in order to add the connection between keyframes, the algorithm performs a *for* loop for everyone of them up until that moment, so the more keyframes the more this loop will go on, but the real problem is the adding of the connection between this important frames, since they are very near, the angle view is small, a lot of them, connection point, are present, and so, the algorithm take a lot of time to find them all. Then, the third culprit is the creation of new map points, the creation is done through triangulation of the points of the current keyframe and the ones of the set, but it excludes automatically the one that have an angle view too small. The other times are almost linear, without any points of interest, apart the time for the reconstruction of the camera pose in Fig. 6.19 where a spike is present, this is due to the low number of features matched used for its reconstruction, as can be seen from the figure on the right. Up until the point at which the algorithm was forcefully stopped, the error was not so big, having an RMSE of 1.4152 m but, since the objective of this thesis is to check if a real time implementation of the V-SLAM method could be possible, this results have been discarded.

Increasing the FS, the times keep decreasing in a linear manner while the error, more specifically, its RMSE, oscillates but in a general view, it keeps decreasing too. Arriving at FS equal to 7 the maximum time needed for the algorithm goes below two seconds and the reconstruction error is also minimized, with an RMSE of 1.662 m. The results for this FS can be seen in Figs. 6.20, 6.21, 6.22.

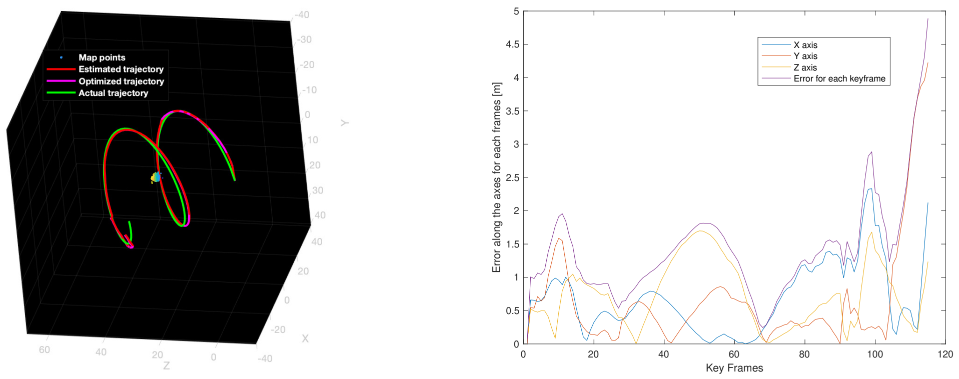


Figure 6.20: *7 FS: Comparison between the reconstructed and real trajectory (left), Error (right)*

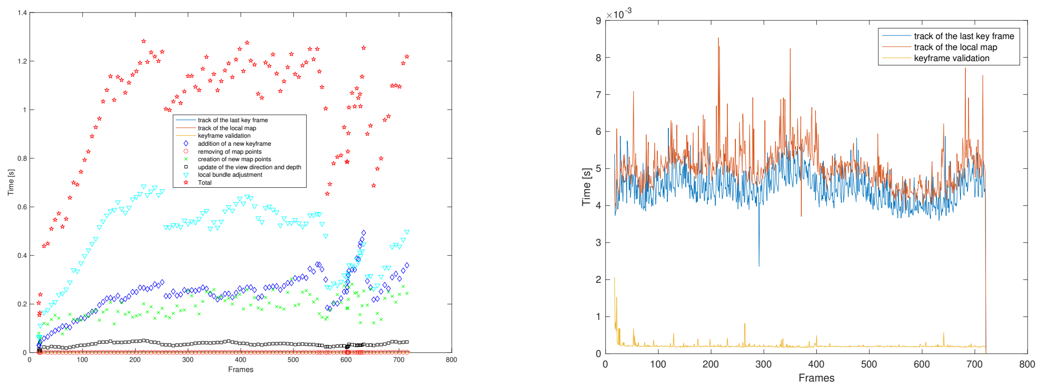


Figure 6.21: *7 FS: Important times during the tracking (left), closeup to the first three important times that are not visible (right)*

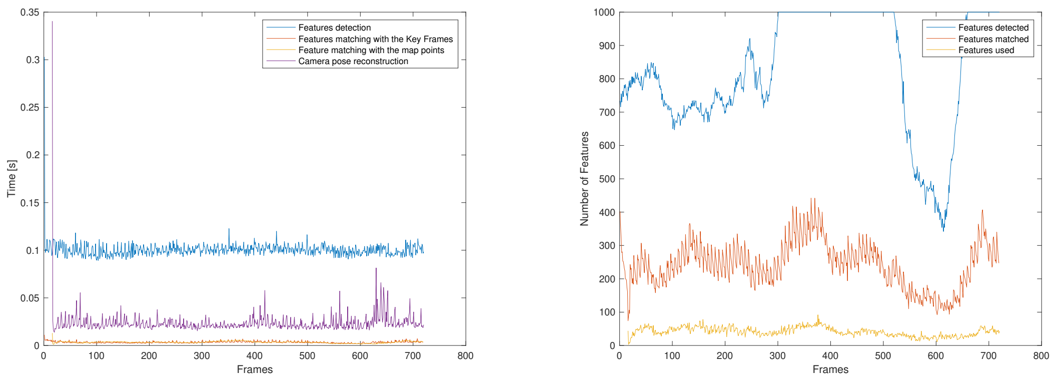


Figure 6.22: *7 FS: Fundamental times (left), Number of features (right)*

It can be seen that the maximum time necessary by the algorithm is a little more than 1.2 seconds and that the majority of the results stay between one and 1.2 seconds, making the real time implementation of it a possibility. Again, as the 2 FS case, the sections of the algorithm that take the most time are the local bundle adjustment, the addition of a new keyframe and the creation of new map points. From Fig. 6.20 and Fig. 6.22 an interesting consideration can be extrapolated, the error starts increasing when the number of features detected, matched and used start decreasing, in fact, it can also be seen in the trajectory that from that point the reconstructed trajectory starts going adrift.

Instead, using a FS equal to 8, the time is minimized, while the error is a bit higher, having an RMSE of 2.757 m. In Figs. 6.23, 6.24, 6.25 the result for this simulation are illustrated.

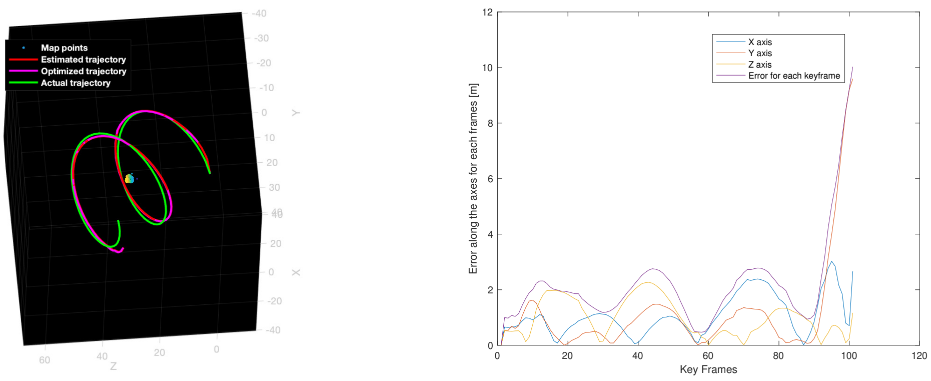


Figure 6.23: *8 FS: Comparison between the reconstructed and real trajectory (left), Error (right)*

Here, the highest time needed by the algorithm is a little more than 1.1 seconds and the majority of the cases stay between 0.8 and 1 second, which is better than before but, as previously said, the RMSE of the error is bigger. So, this time, going from FS equal to 7 to FS equal to 8, a reduction in the algorithm's run-time is present but it is paid by a bigger error. Again, the sections of the algorithm that take the most running-time

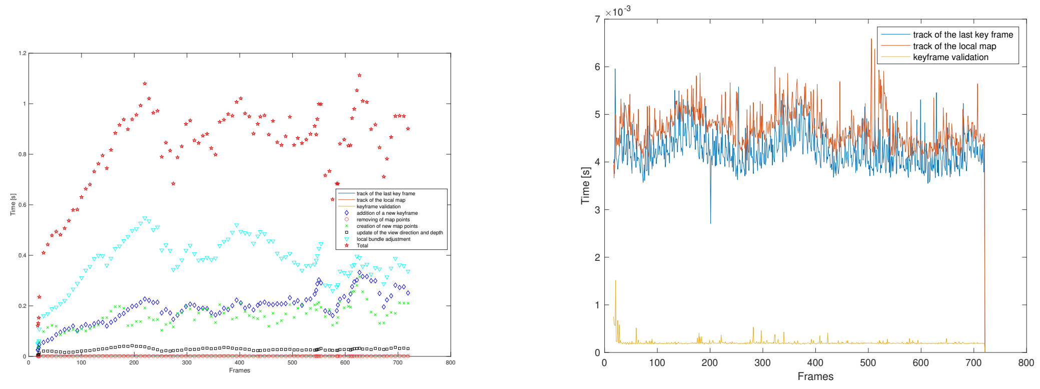


Figure 6.24: *8 FS: Important times during the tracking (left), closeup to the first three important times that are not visible (right)*

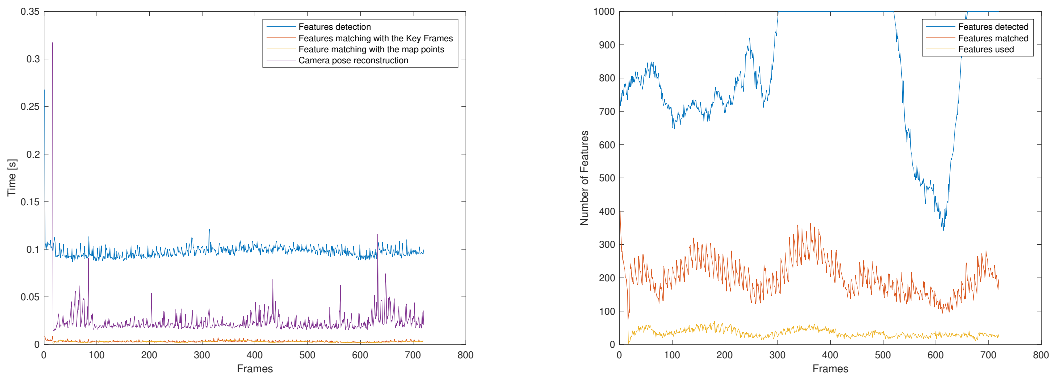


Figure 6.25: *8 FS: Fundamental times (left), Number of features (right)*

are the same three as before, but, due to the reasons explained before, it doesn't come as a surprise anymore. Another interesting fact is that also here the error starts going adrift when the number of features detected, matched and used decrease, in fact before that event, the error is comparable to the previous case, being just a bit higher, but not by much. Obviously, if the possibility of a real time implementation could be exploited before, now it is even better, at least for what regards the time side.

Since for two cases the errors started increasing at the same point, a further investigation has been performed, it uses all the FS up to 8, excluding FS equal to two since the times were too high. The results of this investigation is that for all the cases studied the trajectory starts going adrift exactly at the same point so, intrigued by this fact, it has been checked what happens in those frames and why the features detected decrease. It has been observed that in those frames the only source of light is the one of the Earth and shadow zone are also present.

6.3.2.2 Half trajectory

In the previous section it has been said that in order to overcome the limitation of the map initialization, two things could be done, increasing the angle view or changing the initial position in which the algorithm will start. Here this exact case is shown, in fact it has been decided to let the algorithm work starting from midway so, only half of the trajectory will be mapped, to be more precise, the last half of it.
In this case, the map is initialized with the first and second frame, in Fig. 6.26 and Fig. 6.27 the first frame and the matching for the initialization are reported.

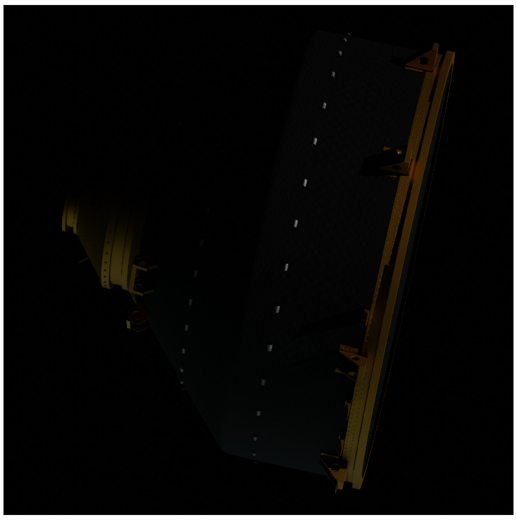


Figure 6.26: *First frame*

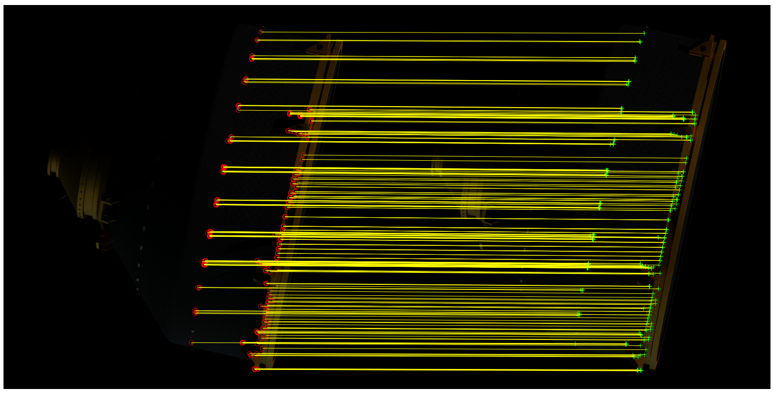


Figure 6.27: *Map initialization matching*

From Fig. 6.26 the separation between the light coming from the Earth and the one coming from the Sun can be appreciated.

In Fig. 6.28 it can be seen that the algorithms stops prematurely and the full trajectory is not reconstructed, this time not because the author stopped it due the very high run-time but because an error occurs due to the fact the the algorithm is not able to map correctly the map points due to the nearness, in terms of angle view, of the keyframes. Even if stopped, it can already be seen that the reconstruction error is quite big although the map initialization is performed in a relatively good way and the relative position of the camera is right.

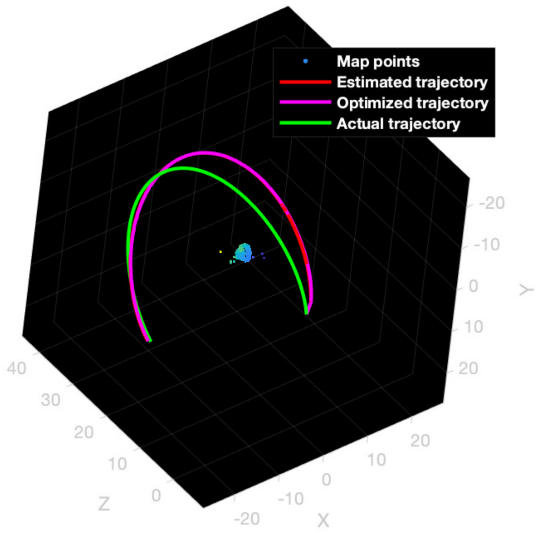


Figure 6.28: *Comparison between the reconstructed and real trajectory with 1 frame-step.*

For these reasons the first frame-step analyzed is 2. The results are shown in the figures below, Figs. 6.29, 6.30, 6.31

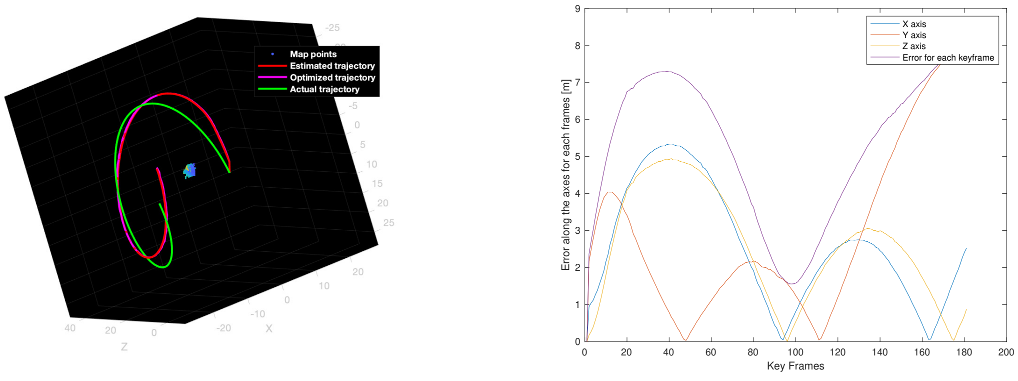
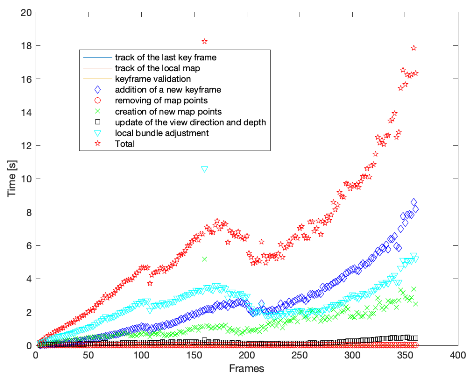


Figure 6.29: *2 FS: Comparison between the reconstructed and real trajectory (left), Error (right)*

From these figures it can be seen that the reconstruction error is quite big but this is not even the worst result, as before, the time needed to process each keyframe is very high,



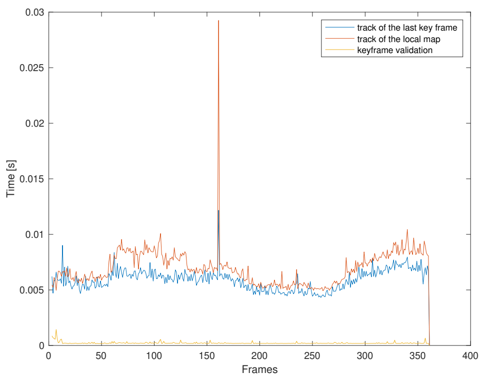
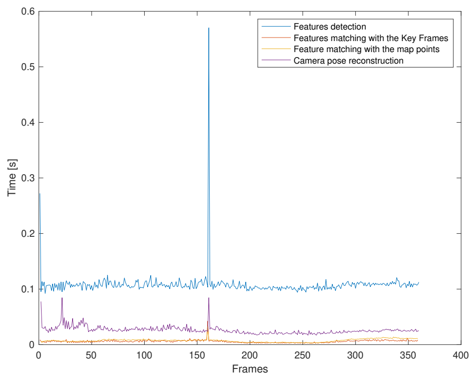


Figure 6.30: *2 FS: Important times during the tracking (left), closeup to the first three important times that are not visible (right)*



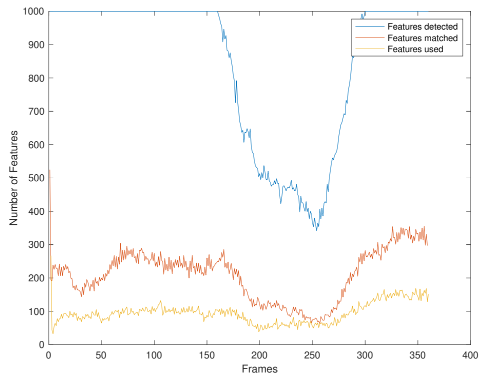


Figure 6.31: *2 FS: Fundamental times (left), Number of features (right)*

making the real time vision-based relative navigation impossible to be performed. Again, the three main sections responsible are local bundle adjustment, keyframe addition and creation of new map points, but in this case, since it was not stopped prematurely, the time needed by the keyframe addition section becomes higher than the one of the local bundle adjustment. The RMSE for the trajectory in this case is 5.65 m.

As for the case before, increasing the FS, the error decreases, reaching its minimum at the FS equal to 8, while the run-time is at its minimum at FS equal to 7. So, going directly to the results of frame-step=7 and frame-step=8, respectively Figs. 6.32, 6.33, 6.34 and Figs. 6.35, 6.36, 6.37, it can be seen the the execution time for the first, 7 FS, is lower than the execution time of the second, but the RMSE for the trajectory of the frame-step=8 case is 3.98 m while for the frame-step=7 case is 4.45 m. Although lower than the frame-step=2 case, the RMSE is still a bit too high to enable vision-based relative navigation, especially in close proximity.

Regarding the times instead, they are considerably lower than the previous case and a real time implementation can be performed. In this case, nothing is unexpected, the

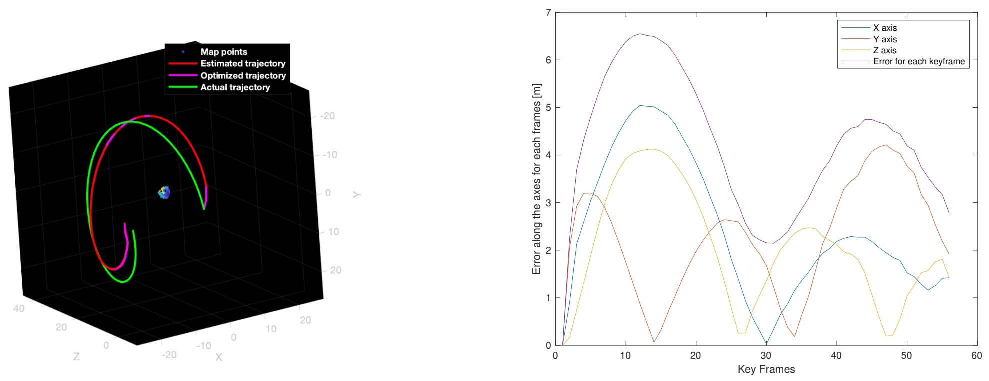


Figure 6.32: *7 FS: Comparison between the reconstructed and real trajectory (left), Error (right)*

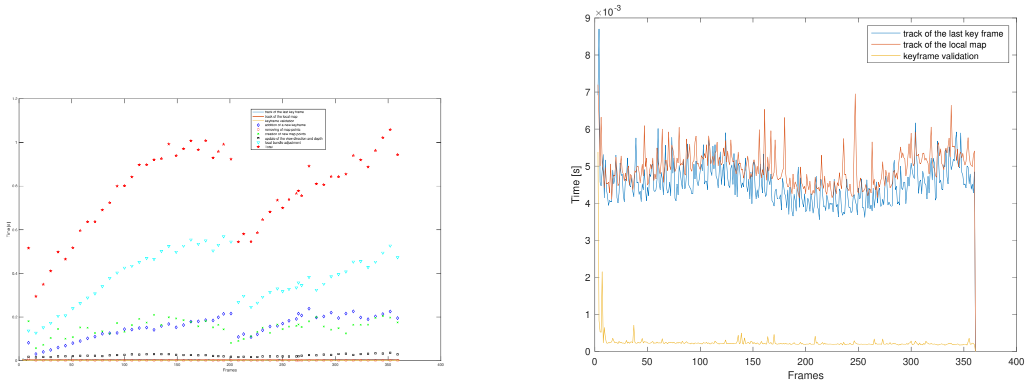


Figure 6.33: *7 FS: Important times during the tracking (left), closeup to the first three important times that are not visible (right)*

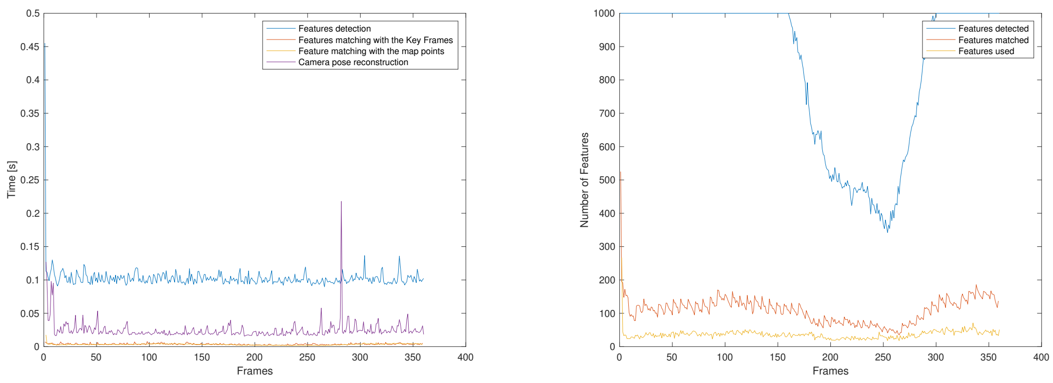
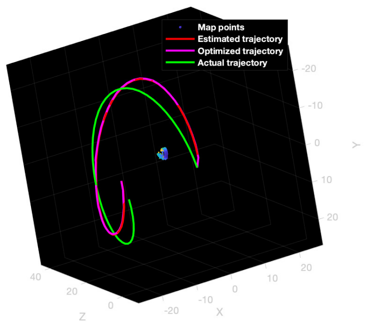


Figure 6.34: *7 FS: Fundamental times (left), Number of features (right)*

bundle adjustment is the main protagonist for the execution time, needing double the time of the second highest time-requiring process.

As can be seen, its true that increasing the FS yields a smaller error but, in general it is still very high, completely different from the benefits provided by a good map initialization as seen in Section 6.3.2.1.



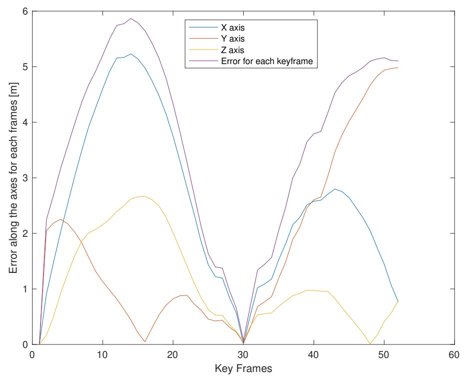
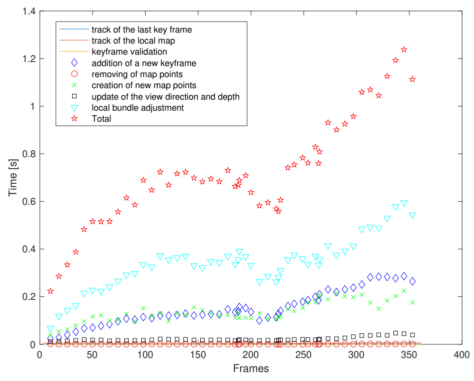


Figure 6.35: *8 FS: Comparison between the reconstructed and real trajectory (left), Error (right)*



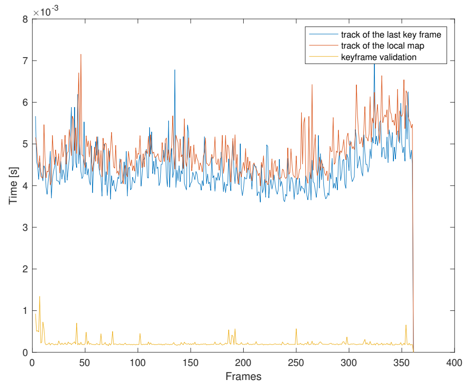
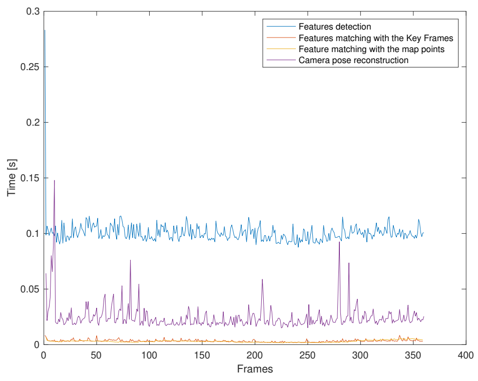


Figure 6.36: *8 FS: Important times during the tracking (left), closeup to the first three important times that are not visible (right)*



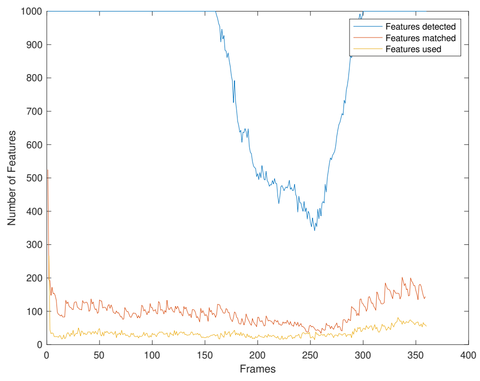


Figure 6.37: *8 FS: Fundamental times (left), Number of features (right)*

Following this line of thought, it has been decided to try to increase also the angle view for the map initialization. Since before, by increasing it, the results became better and better, this was the expected outcome of this trial, but in reality it is the contrary. The following consideration has been made by taking into account all the possible FS, up to 8, for all the tried angle view.

By increasing the angle view the results not only don't become better but they keep getting worse and worse, in fact what can be thought as a good reconstruction for the map initialization at first, become immediately a source of problem for the reconstruction of the following step. In Fig. 6.38 the reconstructed trajectory with the map initialization performed by the first and third frames and a FS of 2 is shown. As can be seen with this angle view at first the reconstructed trajectory follows the real one but it goes adrift really soon.

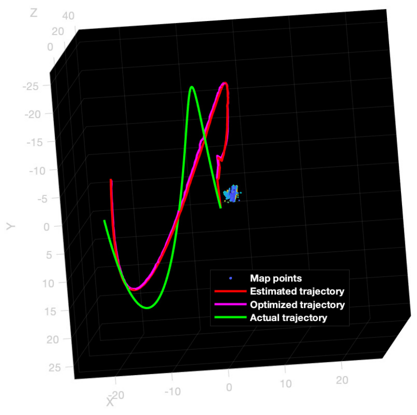


Figure 6.38: *2 FS: Comparison between the reconstructed trajecotry and the real one with the map initilization performed by the first and third frame.*

Increasing the angle view, performing the map initialization with the first and fifth frames, and using an FS equal to 8, the results obtained are shown in Fig. 6.39

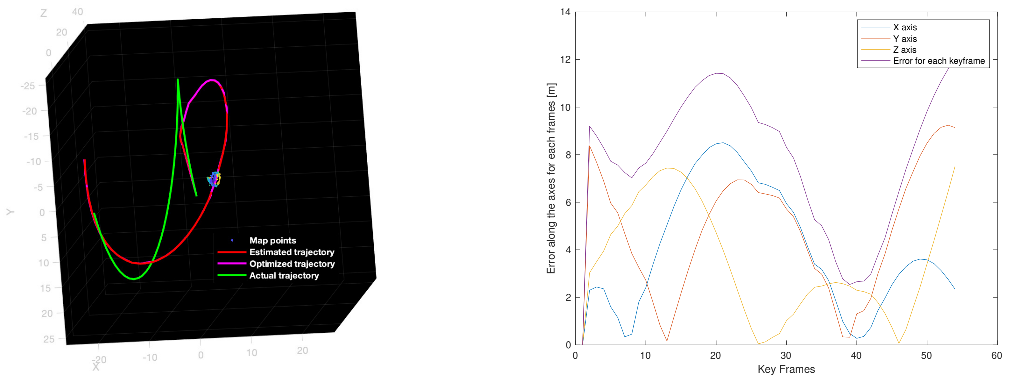


Figure 6.39: *8 FS: Comparison between the reconstructed trajectory and the real one with map initialization performed by the first and fifth frame (left), error (right)*

Looking only at the trajectory it could be thought that the map initialization is performed in a good way but, adding the view of the error, it can be seen that even that step is not performed in a good way, in fact, the error is very high, especially the one along the Y axis, pointing at a reconstruction far above the real one. For the rest of the trajectory it can be immediately noticed that is far from being accurate.

The reason for this behavior, not only the error becoming worse with an increasing angle view but also the fact that in general the error is quite big, should be traceable to a property of VESPA, its axisymmetric body. Since the starting point is almost from its side, this property is put immediately into action and prevents the algorithm to reconstruct the trajectory in an accurate way. This didn't happen for the full trajectory case since when going to its side the map was already initialized in a good way and the algorithm could proceed without much problems.

Just from these two cases it can be seen how changing the initial condition of the algorithm, in terms of frames, change also the end results, validating the statement made at the beginning of the section where, although good, or bad, the results stand true only for their case and, even if some general consideration can be extrapolated, they must be further on verified with a large number of tests.

6.3.2.3 No local bundle adjustment

Here, the local bundle adjustment of the algorithm, the one that in previous images was shown to take the highest execution time, is excluded. The expected results are a decrease in the accuracy of the reconstructed trajectory and a lower execution time.

The cases studied are the one present in the previous sections, so half trajectory with FS equal to 7 and 8, and the full trajectory. In Fig. 6.40, 6.41, 6.42 the results for the half trajectory case with FS equal to 8 can be seen.

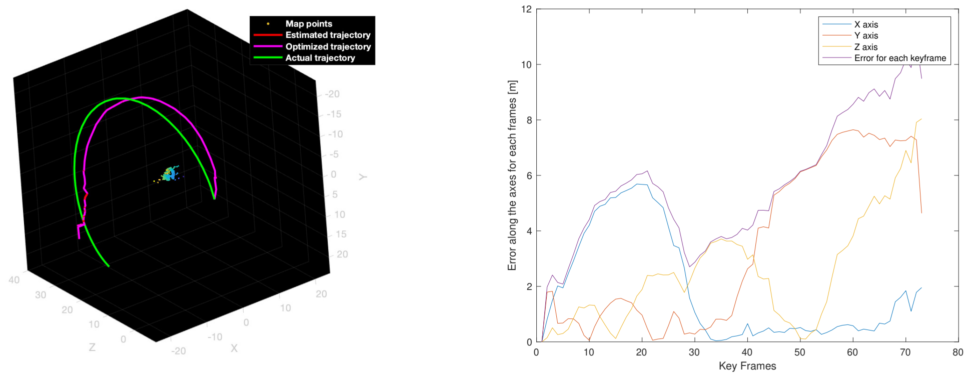


Figure 6.40: *8 FS: Comparison between the reconstructed and real trajectory (left), Error (right)*

From Fig. 6.40 it can immediately seen that the trajectory is not reconstructed completely and this is due to the fact that, as shown in Fig. 6.42, the number of matched features goes to zero and so reconstruction is no longer possible. Other two things that can be noticed are the, as expected, reduction in the execution time, Fig. 6.41, and decrease of the accuracy, Fig.6.40, the RMSE is in fact 6.1134 m. Going back to the times, although lower in general, it can be seen as the time required to add a new keyframe is higher with

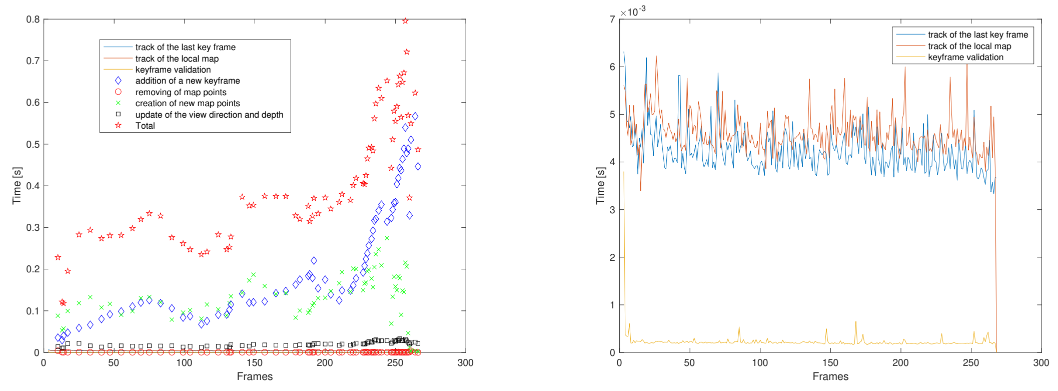


Figure 6.41: *8 FS: Important times during the tracking (left), closeup to the first three important times that are not visible (right)*

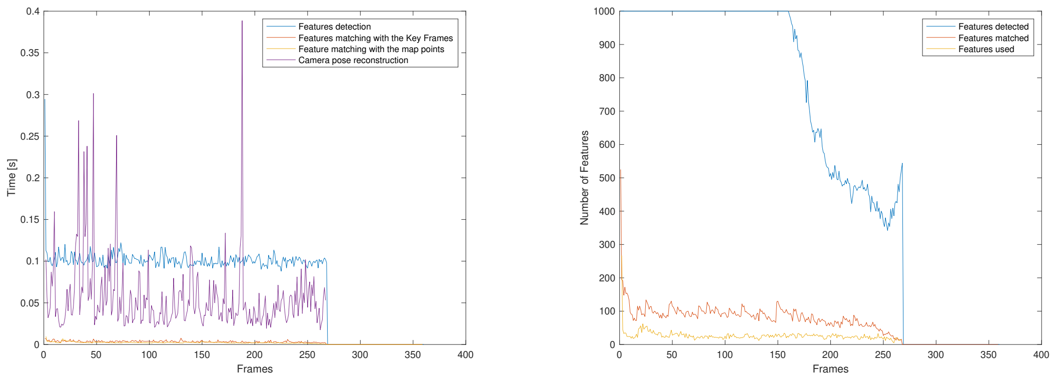


Figure 6.42: *8 FS: Fundamental times (left), Number of features (right)*

respect to the case with local bundle adjustment.

Talking about the case with frame-step=7, the results are shown in Fig. 6.43, 6.44, 6.45.

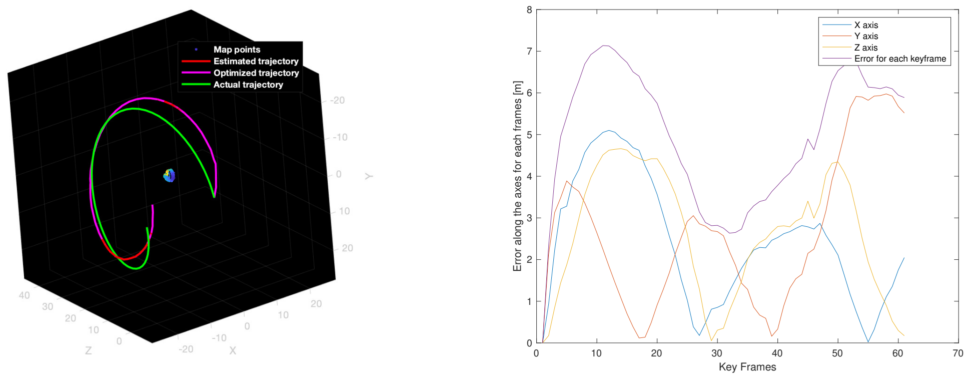


Figure 6.43: *7 FS: Comparison between the reconstructed and real trajectory (left), Error (right)*

Here the trajecotry is fully reconstructed and better global results are obtained with respect to the frame-step=8 case above. Instead, comparing it to the case with local bundle

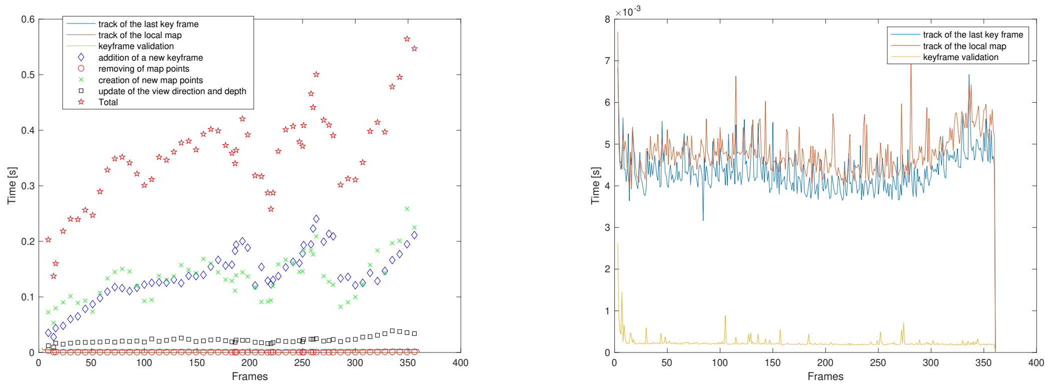


Figure 6.44: *7 FS: Important times during the tracking (left), closeup to the first three important times that are not visible (right)*

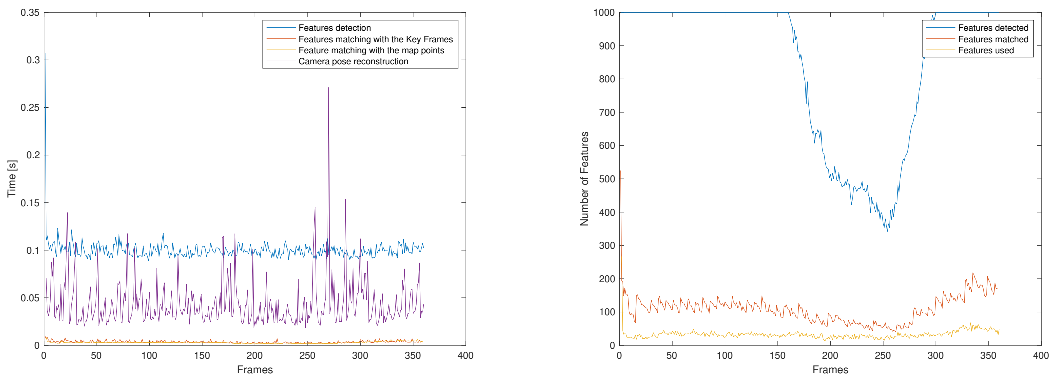


Figure 6.45: *7 FS: Fundamental times (left), Number of features (right)*

adjustment, it's easy to see that the execution time is almost halved and the accuracy is worse, with an RMSE of 5.2384.

In the case of the full trajectory, starting with an FS equal to 8, the algorithm runs until the last frame but what is reconstructed can't be called a trajectory, pointing out the importance of the local bundle adjustment performed at the last step of the algorithm. Then going to FS equal to 7, the trajectory isn't fully reconstructed since problems arise in the transformation from world points to image points, since no world points are found. The same thing happens for FS equal to 6, just that the algorithm stops further on along the trajectory. Then an FS of 5 is used, but as for FS equal to 8, the trajectory reconstructed can't be called a trajectory anymore, the only difference is that for at least half of it, it is reconstructed in a good way while, for the FS of 8, it started to go adrift immediately. All this results points out the importance of the local bundle adjustment step. Finally, going to FS equal to 4, the trajectory is fully reconstructed and in a good way. In Figs. 6.46, 6.47, 6.48 the comparison between the case with the local bundle adjustment, on the left, and without, on the right, can be appreciated.

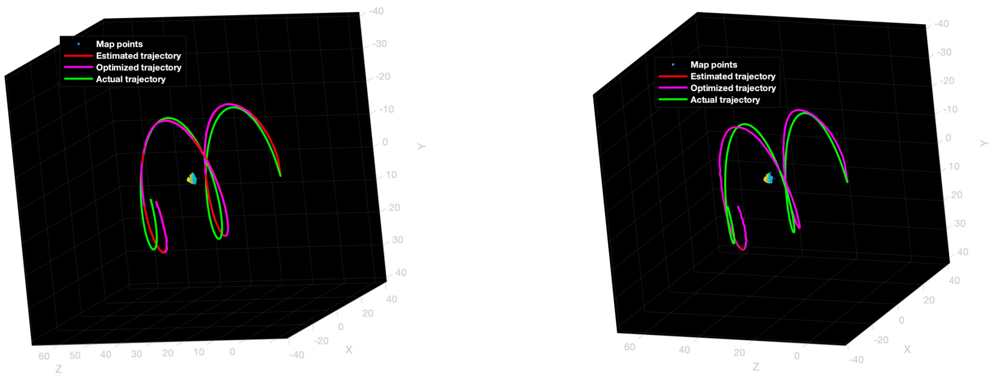


Figure 6.46: *4 FS: Comparison between the reconstructed and real trajectory.*

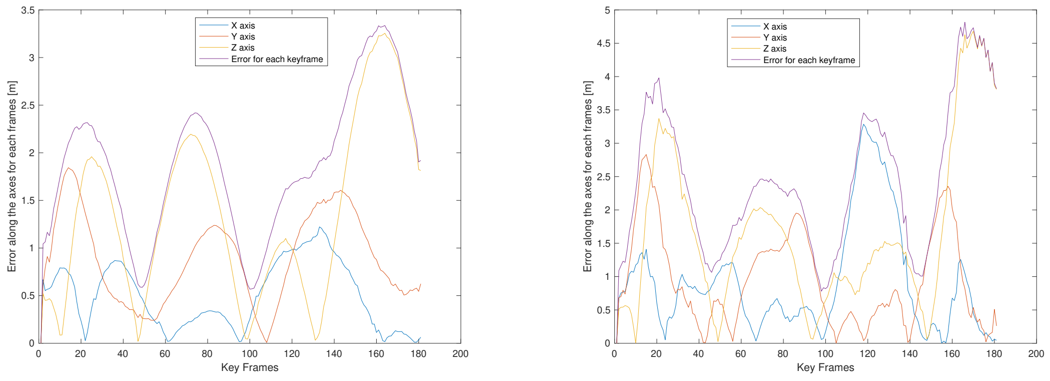


Figure 6.47: *4 FS: Error.*

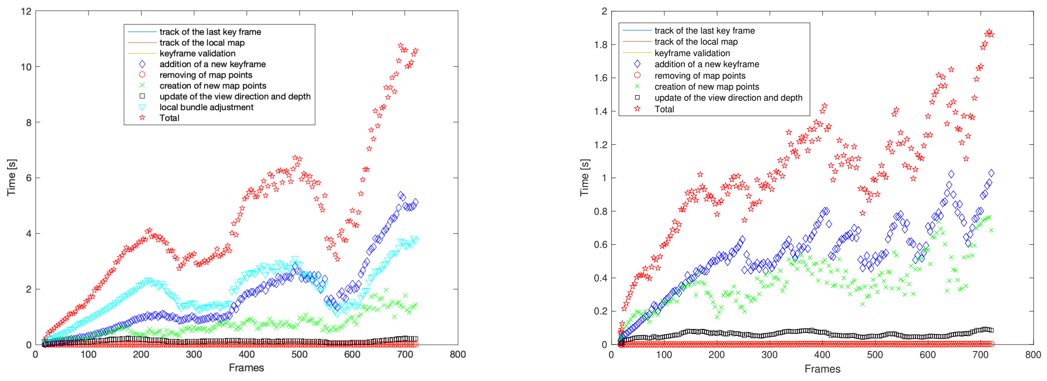


Figure 6.48: *4 FS: Important times during the tracking.*

The reconstructed trajectory and the error are not so different, with, as expected, the case with local bundle adjustment having a smaller one, but, the times are worlds apart, almost a full order of magnitude less for the case without bundle adjustment. The RMSE for the first is 2.0185 while for the second it is 2.6023. For this case it almost seems like that the local bundle adjustment is a weight bringing down the performances of the algorithm, but this is just a single positive case with respect to all the other ones that confirm its importance.

It could be said that without the local bundle adjustment it is better to use smaller FS since more information can be retrieved this way, being always careful to not use too small ones since they could increase again the time needed for the execution of the algorithm, in fact an optimum here was found for FS equal to 4, going lower, the times will start increasing again.

Even though for the FS equal to 4 the addition of the local bundle adjustment is not such a good thing, as it is visible from the previous results, at least for this specific test case 1, the case of optimum FS with local bundle adjustment gives better results than the one of optimum FS without it.

6.3.3 Test case 2

Now, in order the bypass the axysimmetric body of VESPA, its z axis is positioned perpendicular to the velocity direction. With this configuration, unlike the previous case, no problem arise. In Fig. 6.49 the first frame and in Fig. 6.50 the map initialization matching for this case are presented. The map is initialized with the first and second frame.

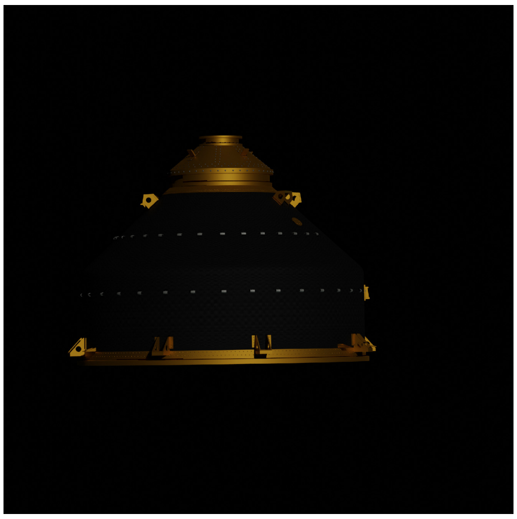


Figure 6.49: *First frame*

Again, using a low frame-step gives worst results than using an high one. In this case only an optimum frame-step exist, minimizing both the execution time and the trajectory error, it is frame-step equal to 9. In Figs. 6.51, 6.52, 6.53 the results given by such a frame-step can be seen. It is immediately noticed that the reconstructed trajectory in this case is way better than in the previous one, in fact the RMSE is 0.91686 m. Also the times are very satisfactory, with the highest one being around 1.3 second and the

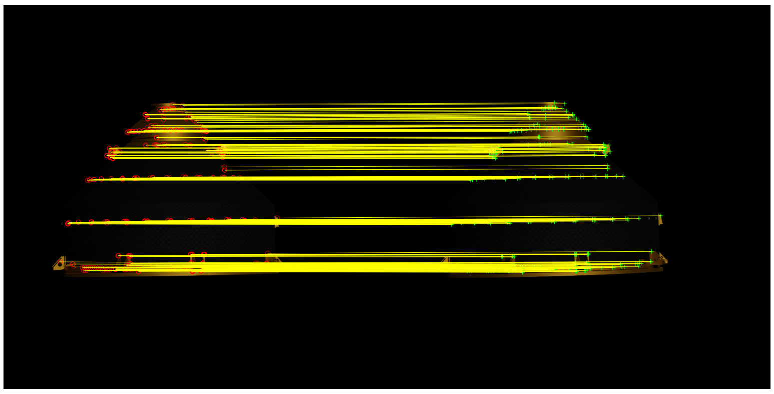


Figure 6.50: *Map initialization matching*

majority staying in the range from 0.8 to 1.2 second and a real time implementation could be possible.

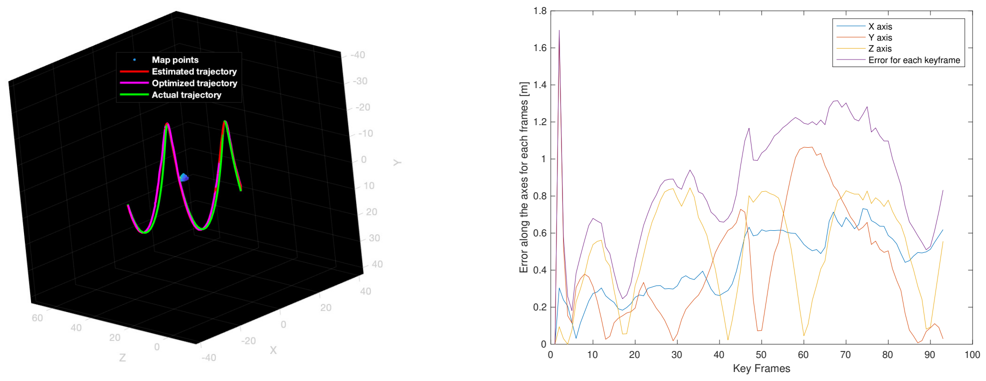


Figure 6.51: *Comparison between the reconstructed trajectory and the real one (left), Error (right)*

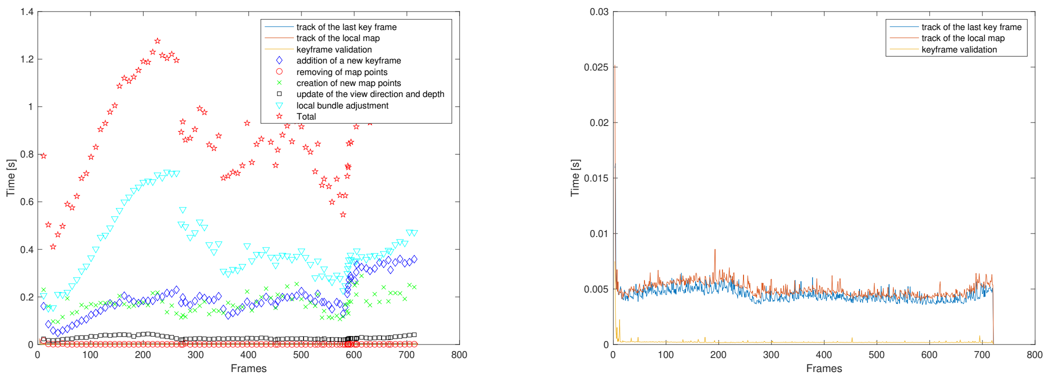


Figure 6.52: *Important times during the tracking (left), closeup to the first three important times that are not visible (right)*

From Fig. 6.53 it can be seen that the fewer the matched feature, and consecutively the one used for the camera pose reconstruction, the higher the time needed for the camera

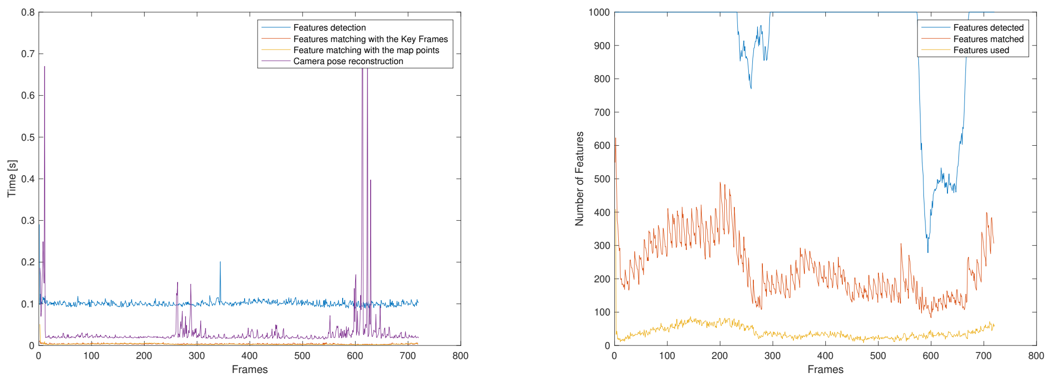


Figure 6.53: *Fundamental times (left), Number of features (right)*

pose reconstruction. Those points are in correspondence of the passage of the camera under VESPA, where few features are present due to the fact that no information about its interior is known and so it has been modelled as completely void of any particular element, having only the carbon fiber texture.

Summarizing, it is evident that having VESPA with this orientation is beneficial to the trajectory reconstruction, although the times are comparable to the ones of Test case 1, the RMSE is way better.

6.3.3.1 No local bundle adjustment

Even for this case, tests with no bundle adjustment were performed. For the precedent optimum frame-step the trajectory is not fully reconstructed and the reconstruction error, until the algorithm stopped, was bigger. The execution time was lower but since problems arise and the algorithm doesn't end, they have no meaning. To let the algorithm run until the last frame a frame-step equal to 7 must be used.

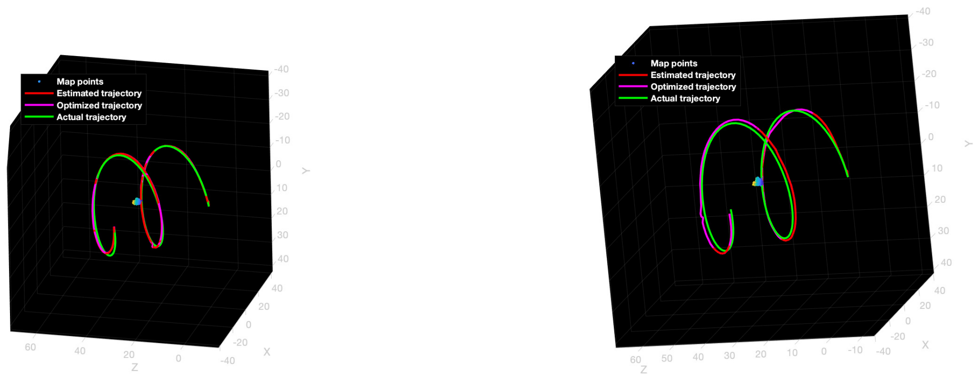


Figure 6.54: *7 FS: Comparison between the reconstructed trajectory and the real one*

The results are not unexpected, the trajectory error is bigger and the execution times lower with respect to the case with local bundle adjustment. In Figs. 6.54, 6.55, 6.56

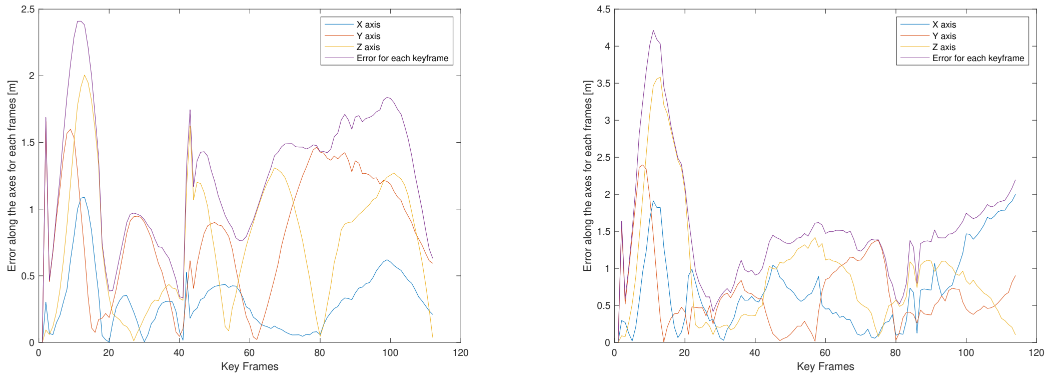


Figure 6.55: *7 FS: Error*

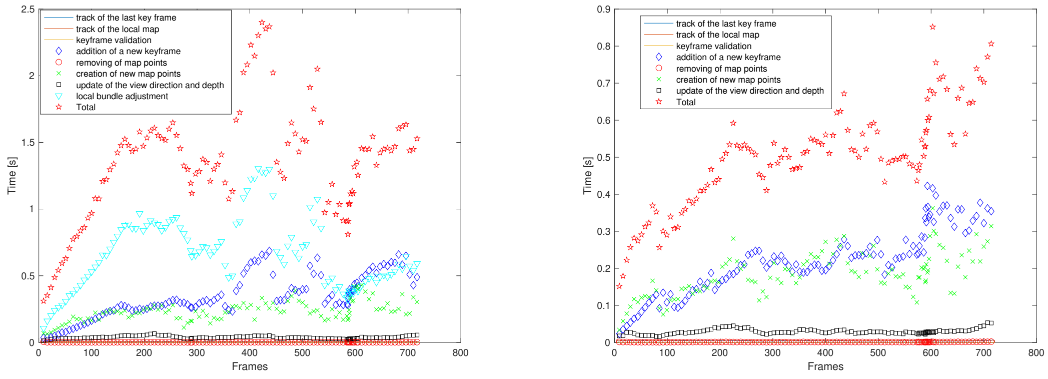


Figure 6.56: *7 FS: Important times during the tracking*

the case with, on the left side, and without, right side, the local bundle adjustment, are reported, respectively, to let the reader have a visible paragon. The RMSE for the case with the local bundle adjustment is 1.35 m while without it is 1.724 m.

6.3.4 Test case 3

Starting from the Test case 2, since it is the one with the better results, an image of the Earth, Fig. 6.57, has been used to create a background. This has been done since the Earth introduce some disturbances in the entire algorithm making the reconstruction of the trajectory more difficult and less accurate, this is due to the fact that some feature will be detected, described and matched also for it, creating an interference with the ones of VESPA.

The trajectory considered consist only of the first 115 frames of the whole trajectory, this is due to the higher rendering time and the fact that this case is performed just to let the reader understand how the background, in any space scenario, affects the results. In Fig. 6.58 the first frame of the sequence can be seen, while in Fig. 6.59, the map initialization step is presented.

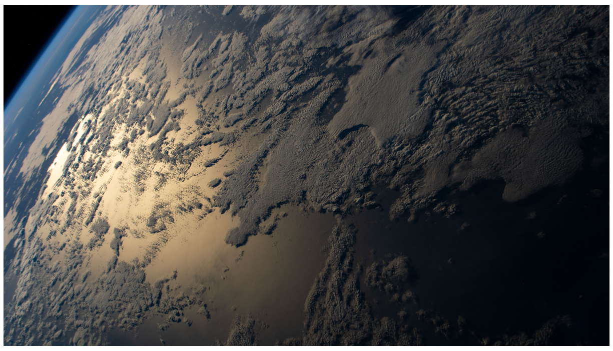


Figure 6.57: *Image of the Earth used as background [67]*

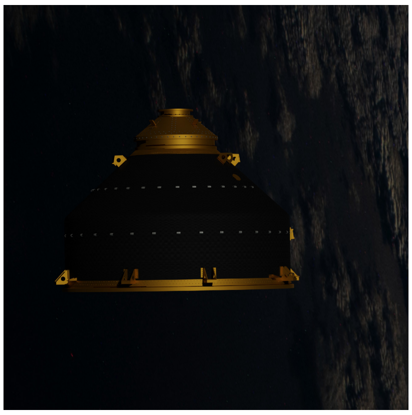


Figure 6.58: *First frame*

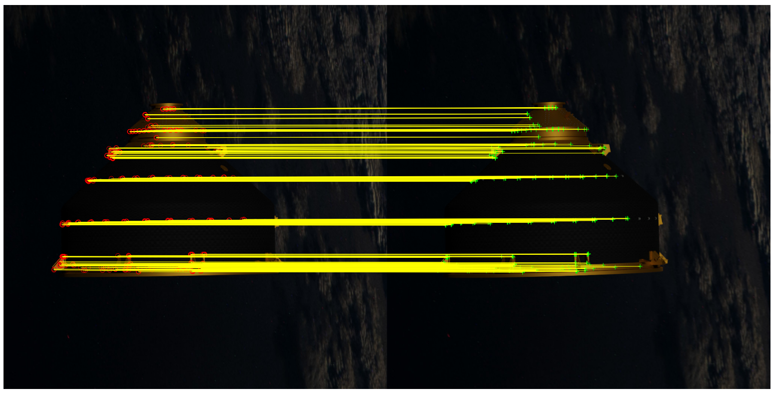


Figure 6.59: *Map initialization matching*

As can be seen from Figs. 6.60, 6.61, with the case having the Earth on the background being on the right side, the reconstructed trajectory is less accurate and so the error bigger, in fact the RMSE of the the Earth-free trajectory is 0.6453 while the other is 1.4452 m, more than two times the first. A comparison between the times is not shown since they are similar and no interesting consideration can be derived from it.

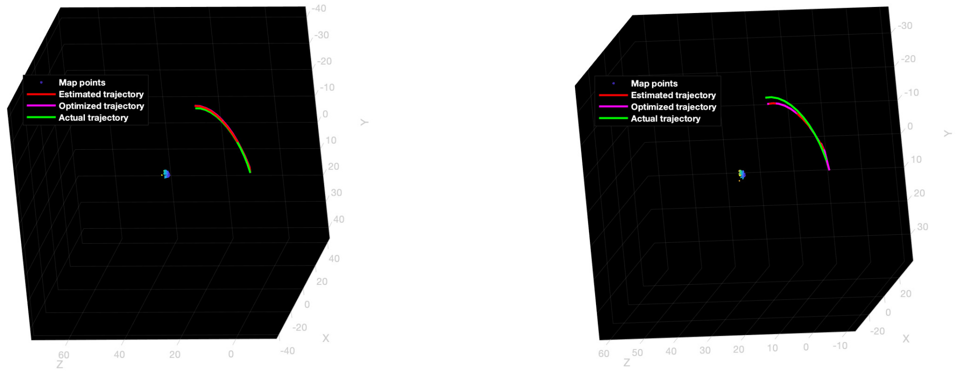


Figure 6.60: *Comparison between the reconstructed trajectory and the real one*

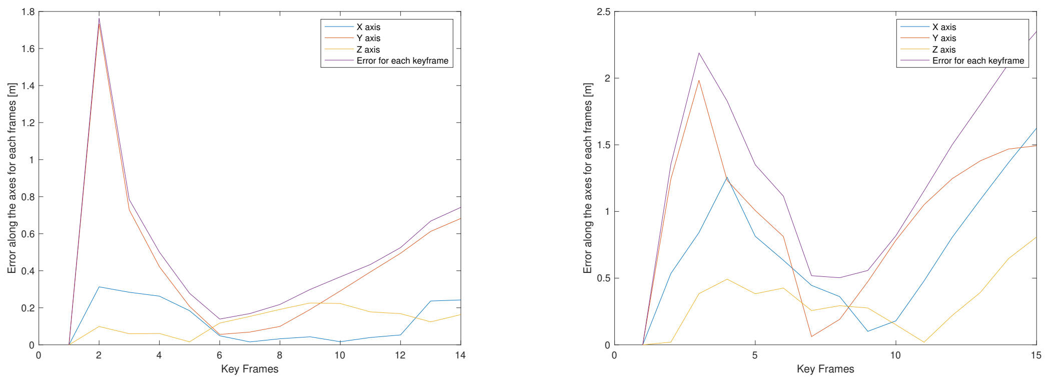


Figure 6.61: *Error*

7 Conclusions

The V-SLAM problem for space application has been presented in this thesis and from the results obtained it is possible to understand the benefits and limitation of such a method.

During the development of this thesis the first hurdle found along the road was the creation of the synthetic images that, although don't belong to the V-SLAM problem, they are the foundations, in this work, in which the problem is built on. The creation of the model was challenging and two main aspects stand out, the first one being the fidelity of the model with respect to the real one, for VESPA, or any object of interest, it is important to replicate even the smallest elements since they would likely be points of interest/features detected by the algorithm. The second one is about the textures used, the better the texture used the more the image will be photorealistic. Then also the environment gave rise to some difficulties, especially in the creation of the background that, as explained, in order to show some results, has been implemented using a flat image due to the low computational power of the used CPU.

Another important brick for the resolution of the V-SLAM problem is the camera calibration, without a good reconstruction of the camera parameters the computer is not able to "see" the image correctly, much likely as a person who wears glasses is not able to see correctly without them or with ones with a wrong gradation, making the detection of features and their consecutive matching very hard.

Going to the algorithm, it is immediately shown the importance of the map initialization step for a good trajectory reconstruction and how its angle view doesn't follow a constant rule but each case has its own one, be it small or large. Particular attention should be given to this step in order to achieve the best results possible. Then also the angle view between each keyframe is taken into consideration and the results seems to point out that a bigger one, in presence of the local bundle adjustment step, gives better results, both in terms of accuracy of the reconstructed trajectory and of time needed for the execution of the algorithm. The limitation of a keyframe approach is shown too, especially regarding the time needed by the addition of a new keyframe in the set. In a long simulation the number of keyframe will be quite high and so that section of the algorithm will likely take more and more time to be executed, making the algorithm unusable. A possible solution could be a window approach that will not take into consideration all the keyframes added up until the new one to search for connection, but only a window around it, this suggestion has been made taking into consideration that after some time the connection between the first keyframes and the current one will be near none.

In conclusion, although the method shows great potential, and will surely play a major role in the future, in the current phase of its study, an integration with data coming from other sensor, like a LiDAR, or another camera, maybe an infrared one in order to retrieve data even during an eclipse, is necessary in order to obtain accurate/satisfactory results.

This analyses, although in a preliminary state, is well aligned with the needs of the incoming space missions interests around small objects, in which the relative navigation has a central role. Active Debris Removal (ADR) missions and asteroid exploration are examples of possible applications, as previously stated. To better evolve the discipline of relative navigation using images, the next section proposes some hints to enhance the work in this thesis.

7.1 Future development

The obtained results are as good as the programs and the procedures used to attain them, so, by improving the latter, also the former will become better. In the sections below, some of the future developments that could be carried out to improve the tools used will be shown.

7.1.1 Synthetic images

Starting from the generation of synthetic images, the target model should be first update with the most recent information and further details could be added. Also the textures can be improved: specific higher quality textures can be used, remembering that one of the main characteristics of a good texture is that of being seamless. But it must be taken into consideration that the higher the texture quality, the more "heavy" it will be and so the rendering time will increase. Other than the target, also the background could be improved. In this work the Earth in the background could not have been applied but, with a more powerful computer, this limitation could be lifted, creating in such a way, a more realistic environment.

A final improvement, and let's say the most simple and immediate one, is changing the image generator software. This could be done by changing completely the software or by upgrading the current one in use with its latest version available.

7.1.2 V-SLAM

In the V-SLAM algorithm the loop closure feature was present, but since the trajectory never passed the same place twice, it was never exploited. In order to see how such a feature works in a space environment an appropriate trajectory could be designed. Another feature that could be added is the failure recovery one.

Another solutions, as for the case of the synthetic images, could be changing the software, in this case ORB-SLAM. As before, two paths can be taken, either improving the current one, a SLAM algorithm is based on a lot of parameters, changing them could give beneficial results, depending on what has been modified and how, or by completely changing the software, using a SURF-SLAM for example.

In order to validate the V-SLAM approach in different kind of scenario, due to the nature of the used tools, it is possible to perform an extensive number of tests, changing something every time, from the camera parameters to the target and the background. Talking about the background, as said before, it is a disturbing element that introduces not useful features in the algorithm. In order to reduce their number, a ROI detection algorithm can be added, in this way the features will be extracted only from the bounded box of the region of interest. It must be remembered that, although beneficial, it will also increase the execution time of the whole algorithm.

Even if simulations through the use of synthetic images are useful, they cannot replace tests in real scenarios with actual sensors and targets. Testing could be conducted on embedded systems or in a GNC facility, such as the one present at Politecnico di Milano with an asteriod and satellite mock-ups.

Bibliography

[1] D. F. Pierrottet F. Amzajerdian, L. B. Petway G. D. Hines, and B. W. Barnes. "Doppler lidar sensor for precision navigation in GPS-deprived environment". In: *Laser Radar Technology and Applications XVIII, Vol. 8731, International Society for Optics and Photonics, SPIE* (May 2013). DOI: 10.1117/12.2018359.

[2] *Capability overview of NASA autonomous systems*. URL: https://www.nasa.gov/sites/default/files/atoms/files/nac_tie_aug2018_tfong_tagged.pdf.

[3] *Tridar, automating a better rendezvous in space*. URL: https://www.nasa.gov/mission_pages/station/research/news/b4h-3rd/it-automating-better-space-rendezvous/.

[4] S. Cryan J. A. Christian. *A Survey of LIDAR Technology and its Use in Spacecraft Relative Navigation*. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2013-4641.

[5] *DART mission*. URL: https://www.nasa.gov/mission_pages/dart/main/index.html.

[6] *ETS-VII mission*. URL: https://global.jaxa.jp/projects/sat/ets7/index.html.

[7] *Hayabusa-2 mission*. URL: https://spaceflight101.com/spacecraft/hayabusa-2/.

[8] *OSIRIS-Rex mission*. URL: https://www.asteroidmission.org/objectives/.

[9] *Rosetta mission*. URL: https://www.esa.int/Science_Exploration/Space_Science/Rosetta/Rosetta_Media_factsheet.

[10] *Rosetta's NavCam instrument*. URL: https://pdssbn.astro.umd.edu/catalogs/Rosetta/navcam_inst.cat.

[11] *Rosetta's OSIRIS instrument*. URL: https://www.mps.mpg.de/1979623/OSIRIS.

[12] Humberto Campins et al. "THE ORIGIN OF ASTEROID 162173 (1999 JU3)". In: *The Astronomical Journal* 146 (2013).

[13] *Bennu*. URL: https://www.minorplanetcenter.net/db_search/show_object?object_id=Bennu.

[14] *OSIRIS-REx instruments*. URL: https://www.asteroidmission.org/objectives/instruments/.

[15] Tim Bailey Hugh Durrant-Whyte. “Simultaneous localization and mapping: Part I”. In: *Robotics and Automation Magazine, IEEE 13* (June 2006), pp. 99–110. DOI: 10.1109/MRA.2006.1638022.

[16] V.Pesce. *Stereovision-based pose and inertia estimation for unknown and uncooperative space objects*. URL: https://www.politesi.polimi.it/handle/10589/107902?mode=complete.

[17] Shai Segal, Avishy Carmi, and Pini Gurfil. “Stereovision-Based Estimation of Relative Dynamics Between Noncooperative Satellites: Theory and Experiments”. In: *IEEE Transactions on Control Systems Technology* 22.2 (2014), pp. 568–584. DOI: 10.1109/TCST.2013.2255288.

[18] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. “ORB-SLAM: A Versatile and Accurate Monocular SLAM System”. In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163. DOI: 10.1109/TRO.2015.2463671.

[19] Hauke Strasdat, J. M. M. Montiel, and Andrew J. Davison. “Real-time monocular SLAM: Why filter?” In: *2010 IEEE International Conference on Robotics and Automation*. 2010, pp. 2657–2664. DOI: 10.1109/ROBOT.2010.5509636.

[20] P. Tsiotras M. Dor. “ORB-SLAM Applied to Spacecraft Non-Cooperative Rendezvous”. In: 2018. DOI: 10.2514/6.2018-1963.

[21] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. “ORB-SLAM: A Versatile and Accurate Monocular SLAM System”. In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163. DOI: 10.1109/TRO.2015.2463671.

[22] F. Fraundorfer D. Scaramuzza. “Visual Odometry [Tutorial]”. In: *IEEE Robotics and Automation Magazine* (Dec. 2011), pp. 80–92. DOI: 10.1109/MRA.2011.943233.

[23] Larry Matthies Yang Cheng Mark Maimone. “Visual odome- try on the Mars exploration rovers”. In: *2005 IEEE International Con- ference on Systems, Man and Cybernetics* 1 (2005), pp. 903–910.

[24] S. M. Parkes et al. “A virtual test environment for validating spacecraft optical navigation”. In: *The Aeronautical Journal (1968)* 117.1197 (2013), pp. 1075–1101. DOI: 10.1017/S000192400000871X.

[25] I. Martin S. M. Parkes. “Virtual lunar landscapes for testing vision-guided lunar landers”. In: *1999 IEEE International Conference on Information Visualization (Cat. No. PR00210)* (1999), pp. 122–127.

[26] *SurRender software*. URL: https://www.airbus.com/space/space-exploration/SurRenderSoftware.html.

[27] *Planet and Asteroid Natural Scene Generation Utility software*. URL: https://www.star-dundee.com/products/pangu-planet-and-asteroid-natural-scene-generation-utility/#technical_specs.

[28] *Blender software*. URL: https://www.blender.org/download/.

[29] RS Wiltshire WH Clohessy. "Terminal guidance system for satellite rendezvous". In: *Journal of the Aerospace Sciences* 27.9 (1960), pp. 653–658.

[30] Vikram Kapila Haizhou Pan. "Adaptive nonlinear control for spacecraft formation flying with coupled translational and attitude dynamics". In: *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No. 01CH37228)* Vol.3 (2001), pp. 2057–2062.

[31] *Describing Orbits*. URL: https://www.faa.gov/about/office_org/headquarters_offices/avs/offices/aam/cami/library/online_libraries/aerospace_medicine/tutorial/media/III.4.1.4_Describing_Orbits.pdf.

[32] *Earth Centered Inertial reference frame*. URL: https://en.wikipedia.org/wiki/Earth-centered_inertial.

[33] Siamak Tafazoli. "On Attitude Recovery of Spacecraft using Nonlinear Control". In: (Dec. 2020), p. 40.

[34] *Perifocal reference frame*. URL: https://adcsforbeginners.wordpress.com/tag/vernal-equinox/.

[35] Hyung-Chul Lim, Hyochoong Bang, and Sangjong Lee. "Adaptive Backstepping Control for Satellite Formation Flying with Mass Uncertainty". In: *Journal of Astronomy and Space Sciences* 23 (Dec. 2006), pp. 405–414. DOI: 10.5140/JASS.2006.23.4.405.

[36] S. R. Vadali K. T. Alfriend, J. P. How P. Gurfil, and L. Breger. *Spacecraft Formation Flying Dynamics, control and navigation*. Elsevier Astrodynamics Series - Butterworth-Heinemann, 2010.

[37] G. Saccomandi P. Biscari T. Ruggeri and M. Vianello. *Meccanica Razionale*. Springer, 2013.

[38] *Perspective*. URL: https://www.tate.org.uk/art/student-resource/exam-help/perspective.

[39] Andrew Zisserman Richard Hartley. *Multiple View Geometry in computer vision*. Cambridge University Press, 2004.

[40] *Parallels lines that intersect at infinity*. URL: https://baweanlov.blogspot.com/2021/03/rette-parallele-si-incontrano.html.

[41] J. Heikkila and O. Silven. "A Four-step Camera Calibration Procedure with Implicit Image Correction." In: *IEEE International Conference on Computer Vision and Pattern Recognition* (1997).

[42] *OpenCV. Camera Calibration and 3D Reconstruction.* URL: https://docs.opencv.org/3.4.4/d9/d0c/group__calib3d.html.

[43] *cause of tangential distortion.* URL: https://uk.mathworks.com/help/vision/ug/camera-calibration.html#buvr2qb-2.

[44] *Tangetial distortion effect.* URL: https://developer.ridgerun.com/wiki/index.php?title=File:Undistort_tangential_distortion_representation.svg.

[45] Wilhelm Burger. "Zhang's Camera Calibration Algorithm: In-Depth Tutorial and Implementation". In: May 2016.

[46] *Lens equation.* URL: https://www.kielia.de/photography/calculator/lens-equation/.

[47] *DSLR camera basics.* URL: https://imaging.nikon.com/lineup/dslr/basics/19/01.htm.

[48] Hasan Joni, Imzahim Alwan, and Ghazwan Naji. "Utilizing Artificial Intelligence to Collect Pavement Surface Condition Data". In: *Engineering and Technology Journal* 38 (Feb. 2020), pp. 74–82. DOI: 10.30684/etj.v38i1A.251.

[49] *Understanding Focal length.* URL: https://www.nikonusa.com/en/learn-and-explore/a/tips-and-techniques/understanding-focal-length.html#.

[50] *Visual navigation for autonomous planetary landing.* URL: http://hdl.handle.net/10589/123100.

[51] Martin A. Fischler and Robert C. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: 24.6 (1981). ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: https://doi.org/10.1145/358669.358692.

[52] *RANSAC algorithm.* URL: https://uk.mathworks.com/discovery/ransac.html.

[53] Chris Kahlefendt. *Implementation and Evaluation of Monocular SLAM for an Underwater Robot.* 2018. URL: https://robotics.ee.uwa.edu.au/theses/2017-UnderwaterSLAM-Kahlefendt.pdf.

[54] Adrian Penate-Sanchez, Juan Andrade-Cetto, and Francesc Moreno-Noguer. "Exhaustive Linearization for Robust Camera Pose and Focal Length Estimation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.10 (2013), pp. 2387–2400. DOI: 10.1109/TPAMI.2013.36.

[55] J. Sturm. *Lecture 6: Visual navigation for flying robots*. URL: https://www.youtube.com/watch?v=HuzC0MAlo0w.

[56] P.Fua V. Lepetit F. Moreno-Noguer. "EPnP: An accurate O(n) solution to the PnP problem". In: *International journal of computer vision 81* (2009). DOI: 10.1007/s11263-008-0152-6.

[57] *VESPA*. URL: https://www.esa.int/Enabling_Support/Space_Transportation/Launch_vehicles/VV02_Vega_uses_Vespa.

[58] *Vega C user manual*. URL: https://www.arianespace.com/wp-content/uploads/2018/%2007/Vega-C-user-manual-Issue-0-Revision-0_20180705.pdf.

[59] *NORAD catalogue*. URL: https://www.norad.mil.

[60] *Blender beginner tutorial*. URL: https://www.youtube.com/watch?v=NyJWoyVx_XI&list=PLjEaoINr3zgEq0u2MzVgAaHEBt--xLB6U.

[61] *Collection of high quality free carbon fiber texture*. URL: http://designbeep.com/2012/01/06/collection-of-high-quality-yet-free-carbon-fiber-texturespatterns-for-designers/.

[62] *f-stop in photography and what it does*. URL: https://www.adobe.com/creativecloud/photography/discover/f-stop.html.

[63] *How to make Earth (Cycles)*. URL: https://www.blenderguru.com/tutorials/earth-cycles.

[64] *Single camera calibration app*. URL: https://www.mathworks.com/help/vision/ug/single-camera-calibrator-app.html.

[65] *Matlab*. URL: https://uk.mathworks.com/products/matlab.html.

[66] J. Sturm et al. "A Benchmark for the Evaluation of RGB-D SLAM Systems". In: *Proc. of the International Conference on Intelligent Robot Systems (IROS)*. Oct. 2012.

[67] *NASA*. URL: https://www.nasa.gov/topics/earth/images/index.html.